

Universitatea Națională de Știință și Tehnologie Politehnica din București
Facultatea de Electronică, Telecomunicații și Tehnologia Informației

Automatic Number Plate Recognition

Proiect Python

Nume student:
Herciu Nichita
Anees Zinati

Grupa:
422G

2024

1. Cerință

Acet proiect are ca scop dezvoltarea unui sistem automatizat pentru recunoașterea numerelor de înmatriculare, utilizând un set de date dedicat. Soluția propusă este capabilă să identifice și să proceseze imaginile sau videoclipurile în care apar plăcuțe de înmatriculare, extrăgând informațiile relevante cu acuratețe și eficiență. Sistemul poate fi utilizat în diverse aplicații, precum monitorizarea traficului, securitatea parcărilor sau gestionarea accesului.

2. Tehnologii utilizate

YOLOv9

YOLO (You Only Look Once) este o familie de modele de detecție a obiectelor, iar YOLOv9 reprezintă o versiune optimizată, cunoscută pentru viteza și acuratețea sa. În acest proiect, YOLOv9 este utilizat pentru a detecta plăcuțele de înmatriculare din imagini sau videoclipuri. Modelul funcționează prin segmentarea rapidă a imaginii și identificarea regiunilor relevante (bounding boxes) care conțin plăcuțele de înmatriculare.

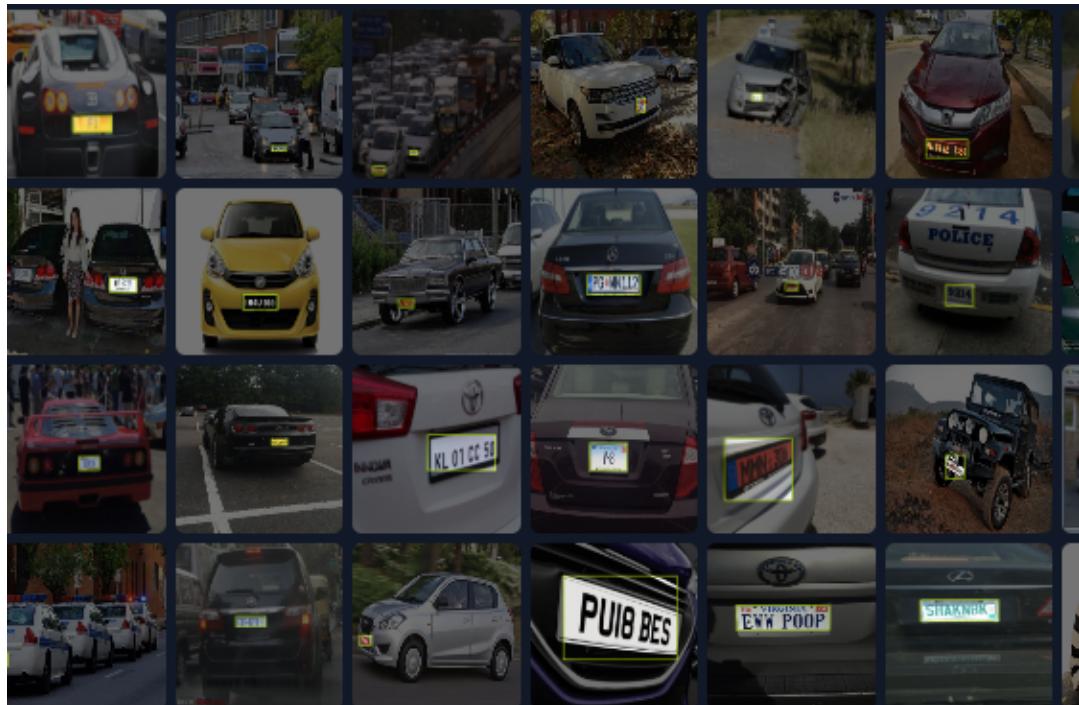




Roboflow

Roboflow este o platformă pentru gestionarea seturilor de date și optimizarea procesului de antrenare a modelelor de învățare automată. În cadrul proiectului, Roboflow a fost folosit pentru:

- Etichetarea și organizarea imaginilor.
- Generarea unui set de date adaptat cerințelor YOLOv9.



EasyOCR

EasyOCR este o bibliotecă simplă și eficientă pentru recunoașterea textului în imagini. După ce YOLOv9 identifică plăcuțele de înmatriculare, EasyOCR este utilizat pentru a extrage textul efectiv de pe plăcuțe. Aceasta suportă o gamă largă de limbi și este foarte potrivită pentru sarcinile de recunoaștere optică a caracterelor (OCR).



Google Colab și GPU NVIDIA

Proiectul a fost implementat și rulat în Google Colab, utilizând GPU-uri NVIDIA pentru a accelera procesul de antrenare și inferență. Colab oferă o platformă accesibilă pentru dezvoltarea proiectelor care necesită resurse computaționale mari.

The screenshot shows a Jupyter Notebook window titled "Python_project.ipynb". The left sidebar displays a file tree with "sample_data", "weights", and "yolov9" folders. The main area contains a terminal window with the command "nvidia-smi" run at 0 sec. The output shows GPU information for a Tesla T4 model, including Persistence-M, Bus-Id, Disp.A, Volatile, Uncorr. ECC, Fan, Temp, Perf, Pwr:Usage/Cap, Memory-Usage, GPU-Util, Compute M., and MIG M. It also lists running processes.

```
[1] !nvidia-smi
Wed Jan 15 09:36:17 2025
+-----+
| NVIDIA-SMI 535.104.05      Driver Version: 535.104.05    CUDA Version: 12.2 |
| Persistence-M | Bus-Id     Disp.A   | Volatile Uncorr. ECC | | | |
| GPU Name      | Fan Temp   Perf    | Pwr:Usage/Cap | Memory-Usage | GPU-Util  Compute M. |
| Fan           | Temp       | Perf          |             |             | GPU-Util  Compute M. |
| Persistence-M | Bus-Id     Disp.A   | Volatile Uncorr. ECC |
| GPU Name      | Fan Temp   Perf    | Pwr:Usage/Cap | Memory-Usage | GPU-Util  Compute M. |
| Fan           | Temp       | Perf          |             |             | MIG M.   |
+-----+
| 0  Tesla T4          Off  00000000:00:04.0 Off   0MiB / 15360MiB | 0%      Default |
| N/A 44C P8          9W / 70W |             0MiB / 15360MiB |             0% | Default |
+-----+
Processes:
| GPU  GI CI PID Type Process name          GPU Memory |
| ID   ID ID   |          |                 Usage |
+-----+
| No running processes found
+-----+
```

3. Descrierea codului

Configurarea mediului

Se verifică disponibilitatea GPU-ului utilizând comanda `nvidia-smi`, esențială pentru accelerarea procesului de antrenare. Apoi, se setează directorul de lucru (`HOME`) pentru a putea organiza fișierele și rula comenziile ulterioară.

The screenshot shows a terminal window with the command "nvidia-smi" run at 0 sec. The output is identical to the one shown in the Jupyter Notebook, displaying GPU information for a Tesla T4 model and no running processes.

```
!nvidia-smi
Wed Jan 15 09:36:17 2025
+-----+
| NVIDIA-SMI 535.104.05      Driver Version: 535.104.05    CUDA Version: 12.2 |
| Persistence-M | Bus-Id     Disp.A   | Volatile Uncorr. ECC | | | |
| GPU Name      | Fan Temp   Perf    | Pwr:Usage/Cap | Memory-Usage | GPU-Util  Compute M. |
| Fan           | Temp       | Perf          |             |             | GPU-Util  Compute M. |
| Persistence-M | Bus-Id     Disp.A   | Volatile Uncorr. ECC |
| GPU Name      | Fan Temp   Perf    | Pwr:Usage/Cap | Memory-Usage | GPU-Util  Compute M. |
| Fan           | Temp       | Perf          |             |             | MIG M.   |
+-----+
| 0  Tesla T4          Off  00000000:00:04.0 Off   0MiB / 15360MiB | 0%      Default |
| N/A 44C P8          9W / 70W |             0MiB / 15360MiB |             0% | Default |
+-----+
Processes:
| GPU  GI CI PID Type Process name          GPU Memory |
| ID   ID ID   |          |                 Usage |
+-----+
| No running processes found
+-----+
```

```
[2] import os  
HOME = os.getcwd()  
print(HOME)  
  
→ /content
```

Clonarea și configurarea YOLOv9

Repo-ul YOLOv9 este clonat pentru a accesa implementarea modelului, iar toate dependențele necesare sunt instalate automat din fișierul requirements.txt.

```
[4] !git clone https://github.com/SkalskiP/yolov9.git  
%cd yolov9  
!pip install -r requirements.txt -q  
  
→ Cloning into 'yolov9'...  
remote: Enumerating objects: 325, done.  
remote: Total 325 (delta 0), reused 0 (delta 0), pack-reused 325 (from 1)  
Receiving objects: 100% (325/325), 2.25 MiB | 4.91 MiB/s, done.  
Resolving deltas: 100% (162/162), done.  
/content/yolov9/yolov9  
----- 1.6/1.6 MB 30.7 MB/s eta 0:00:00
```

Instalarea Roboflow și descarcarea Datasetului

Se instalează biblioteca Roboflow pentru gestionarea datasetului.

Se configerează accesul la workspace-ul și proiectul specific din Roboflow.

Setul de date este descărcat în format compatibil cu YOLOv9 și pregătit pentru utilizare.

```
[8] import roboflow  
  
roboflow.login()  
  
rf = roboflow.Roboflow()  
  
project = rf.workspace("arvind-kumar-wjygd").project("anpr2-syx17")  
version = project.version(8)  
dataset = version.download("yolov9")
```

Descarcarea și Configurarea greutatilor prea antrenate

```
[6] !wget -P {HOME}/weights -q https://github.com/WongKinYiu/yolov9/releases/download/v0.1/yolov9-c.pt  
!wget -P {HOME}/weights -q https://github.com/WongKinYiu/yolov9/releases/download/v0.1/yolov9-e.pt  
!wget -P {HOME}/weights -q https://github.com/WongKinYiu/yolov9/releases/download/v0.1/gelan-c.pt  
!wget -P {HOME}/weights -q https://github.com/WongKinYiu/yolov9/releases/download/v0.1/gelan-e.pt
```

Antrenarea modelului

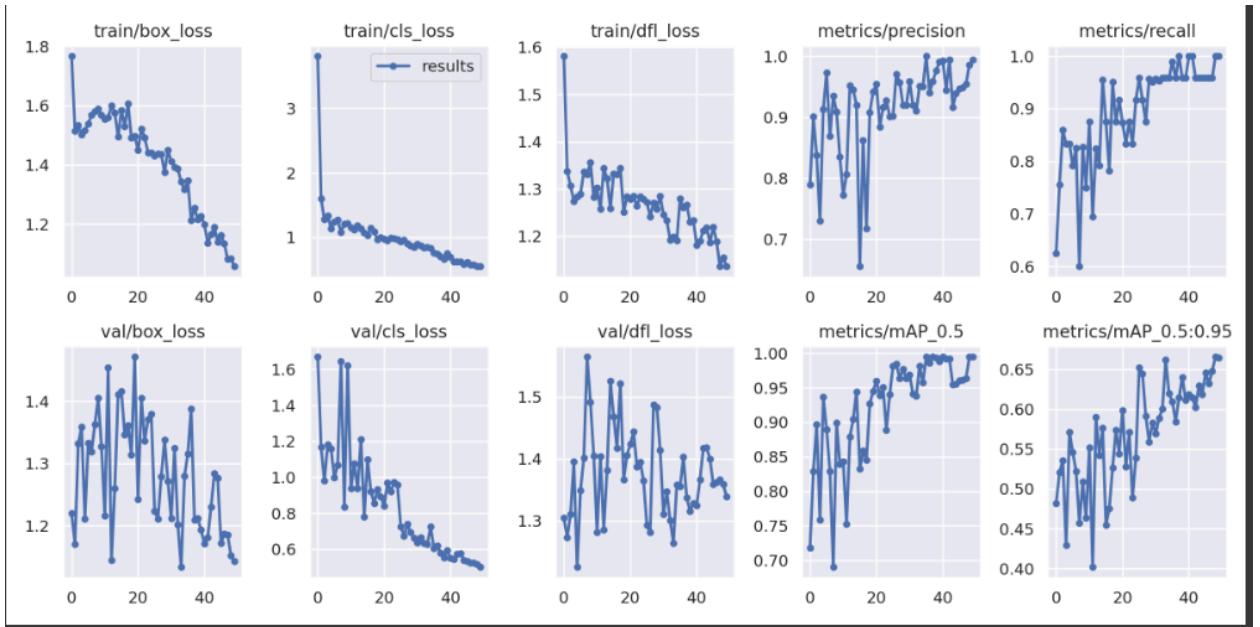
Modelul YOLOv9 este antrenat pe setul de date utilizând parametrii definiți:

- batch: Dimensiunea loturilor pentru actualizarea modelului.
- epochs: Numărul total de epoci de antrenare.
- img: Dimensiunea imaginilor.
- weights: Greutățile pre antrenate utilizate ca punct de plecare.
- data.yaml: Fișierul care definește structura datasetului.

```
[11] %cd {HOME}/yolov9  
  
!python train.py \  
--batch 16 --epochs 50 --img 640 --device 0 --min-items 0 --close-mosaic 15 \  
--data /content/yolov9/yolov9/ANPR2-8/data.yaml \  
--weights {HOME}/weights/gelan-c.pt \  
--cfg models/detect/gelan-c.yaml \  
--hyp hyp.scratch-high.yaml
```

Epoch	GPU_mem	box_loss	cls_loss	df1_loss	Instances	Size
46/49	11.5G	1.135	0.5723	1.189	6	640: 100% 18/18 [00:14<00:00, 1.23it/s]
	Class	Images	Instances	P	R	mAP50 mAP50-95: 100% 1/1 [00:00<00:00, 1.75it/s]
	all	23	24	0.949	0.958	0.961 0.632

Class	Images	Instances	P	R	mAP50	mAP50-95: 100% 1/1 [00:00<00:00, 1.70it/s]
all	23	24	0.987	1	0.995	0.668



Validarea modelului

Modelul este evaluat folosind setul de validare pentru a calcula metricile de performanță (precizie, recall, mAP). Greutățile salvate după antrenare (`best.pt`) sunt utilizate pentru această validare.

```
[ ] %cd {HOME}/yolov9
!python val.py \
--img 640 --batch 32 --conf 0.001 --iou 0.7 --device 0 \
--data {dataset.location}/data.yaml \
--weights {HOME}/yolov9/runs/train/exp3/weights/best.pt
```

Testarea pe imagini noi și vizualizarea rezultatelor

Modelul antrenat este utilizat pentru a detecta plăcuțele de înmatriculare în imagini. Rezultatele sunt salvate într-un director specific ([runs/detect](#)).

```
[ ] !python detect.py \
--img 1280 --conf 0.1 --device 0 \
--weights {HOME}/yolov9/runs/train/exp3/weights/best.pt \
--source /content/yolov9/yolov9/ANPR2-8/test/images
```

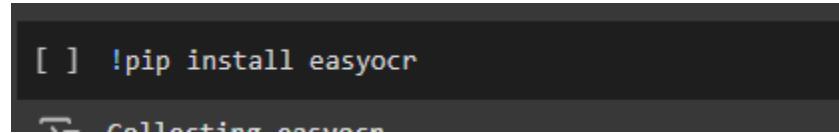


Testarea pe videoclipuri

```
[ ] # video
!python detect.py --conf 0.2 --device 0 --weights {HOME}/yolov9/runs/train/exp3/weights/best.pt --source video1.mp4
⇒ video 1/1 (28/631) /content/yolov9/video1.mp4: 416x640 1 licence, 18.1ms
⇒ video 1/1 (29/631) /content/yolov9/video1.mp4: 416x640 1 licence, 17.9ms
video 1/1 (30/631) /content/yolov9/video1.mp4: 416x640 1 licence, 18.4ms
video 1/1 (31/631) /content/yolov9/video1.mp4: 416x640 1 licence, 17.9ms
video 1/1 (32/631) /content/yolov9/video1.mp4: 416x640 1 licence, 18.1ms
video 1/1 (33/631) /content/yolov9/video1.mp4: 416x640 1 licence, 17.5ms
video 1/1 (34/631) /content/yolov9/video1.mp4: 416x640 1 licence, 19.2ms
video 1/1 (35/631) /content/yolov9/video1.mp4: 416x640 1 licence, 17.7ms
video 1/1 (36/631) /content/yolov9/video1.mp4: 416x640 1 licence, 17.9ms
```

Integrarea OCR

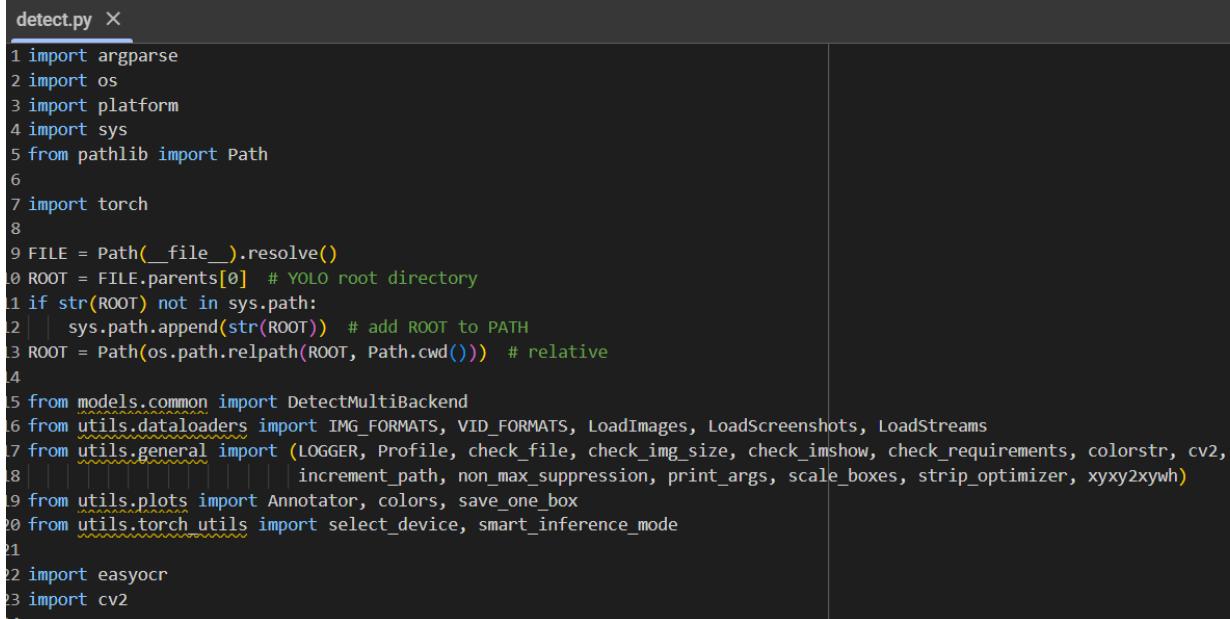
EasyOCR este instalat pentru a extrage textul de pe plăcuțele detectate, adăugând o componentă suplimentară de recunoaștere optică a caracterelor.



```
[ ] !pip install easyocr
[ ] Collecting easyocr
```

4. Partea Python

1. Importul bibliotecilor necesare, configurarea mediului și integrarea yolov9



```
detect.py ×
1 import argparse
2 import os
3 import platform
4 import sys
5 from pathlib import Path
6
7 import torch
8
9 FILE = Path(__file__).resolve()
10 ROOT = FILE.parents[0] # YOLO root directory
11 if str(ROOT) not in sys.path:
12     sys.path.append(str(ROOT)) # add ROOT to PATH
13 ROOT = Path(os.path.relpath(ROOT, Path.cwd())) # relative
14
15 from models.common import DetectMultiBackend
16 from utils.dataloaders import IMG_FORMATS, VID_FORMATS, LoadImages, LoadScreenshots, LoadStreams
17 from utils.general import (LOGGER, Profile, check_file, check_img_size, check_imshow, check_requirements, colorstr, cv2,
18                            increment_path, non_max_suppression, print_args, scale_boxes, strip_optimizer, xyxy2xywh)
19 from utils.plots import Annotator, colors, save_one_box
20 from utils.torch_utils import select_device, smart_inference_mode
21
22 import easyocr
23 import cv2
24
```

2. Integrarea EasyOCR

```
25 reader=easyocr.Reader(['en'], gpu=True)
26
27 def perform_ocr_on_image(img, coordinates):
28     x, y, w, h = map(int, coordinates)
29     cropped_img = img[y:h, x:w]
30
31     gray_img=cv2.cvtColor(cropped_img, cv2.COLOR_RGB2GRAY)
32     results = reader.readtext(gray_img)
33
34     text = ""
35     for res in results:
36         if len(results) == 1 or len(res[1]) > 6 and res[2] > 0.2:
37             text = res[1]
38
39     return str(text)
40
```

Funcția decupează regiunea detectată de yolov9 și extrage textul.

3. Crearea casetelor delimitatoare

```
for *xyxy, conf, cls in reversed(det):
```

```
label = None if hide_labels else (names[c] if hide_conf else f'{names[c]} {conf:.2f}')
annotator.box_label(xyxy, label, color=colors(c, True))
```

4. Functia principala RUN

```
41 @smart_inference_mode()
42 def run(
43     weights=ROOT / 'yolo.pt', # model path or triton URL
44     source=ROOT / 'data/images', # file/dir/URL/glob/screen/0(webcam)
45     data=ROOT / 'data/coco.yaml', # dataset.yaml path
46     imgsz=(640, 640), # inference size (height, width)
47     conf_thres=0.25, # confidence threshold
48     iou_thres=0.45, # NMS IOU threshold
49     max_det=1000, # maximum detections per image
50     device='', # cuda device, i.e. 0 or 0,1,2,3 or cpu
51     view_img=False, # show results
52     save_txt=False, # save results to *.txt
53     save_conf=False, # save confidences in --save-txt labels
54     save_crop=False, # save cropped prediction boxes
55     nosave=False, # do not save images/videos
56     classes=None, # filter by class: --class 0, or --class 0 2 3
57     agnostic_nms=False, # class-agnostic NMS
58     augment=False, # augmented inference
59     visualize=False, # visualize features
60     update=False, # update all models
61     project=ROOT / 'runs/detect', # save results to project/name
62     name='exp', # save results to project/name
63     exist_ok=False, # existing project/name ok, do not increment
64     line_thickness=3, # bounding box thickness (pixels)
65     hide_labels=False, # hide labels
66     hide_conf=False, # hide confidences
```

Defineste parametrii: confidence, weight

Salvează rezultate și ne oferă acele grafice.