Automatic Number Plate Recognition

Overview

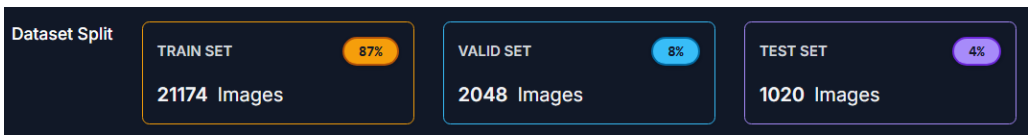
In this project, we utilize the following components for Automatic Number Plate Recognition (ANPR):

- Two YOLOv8 object detectors: One for detecting vehicles and the other for detecting number plates.
- SORT (Simple Online and Realtime Tracking) algorithm for tracking multiple objects in video sequences.
- EasyOCR for extracting text from the detected number plates.

1 license_plate_detector.pt

Dataset

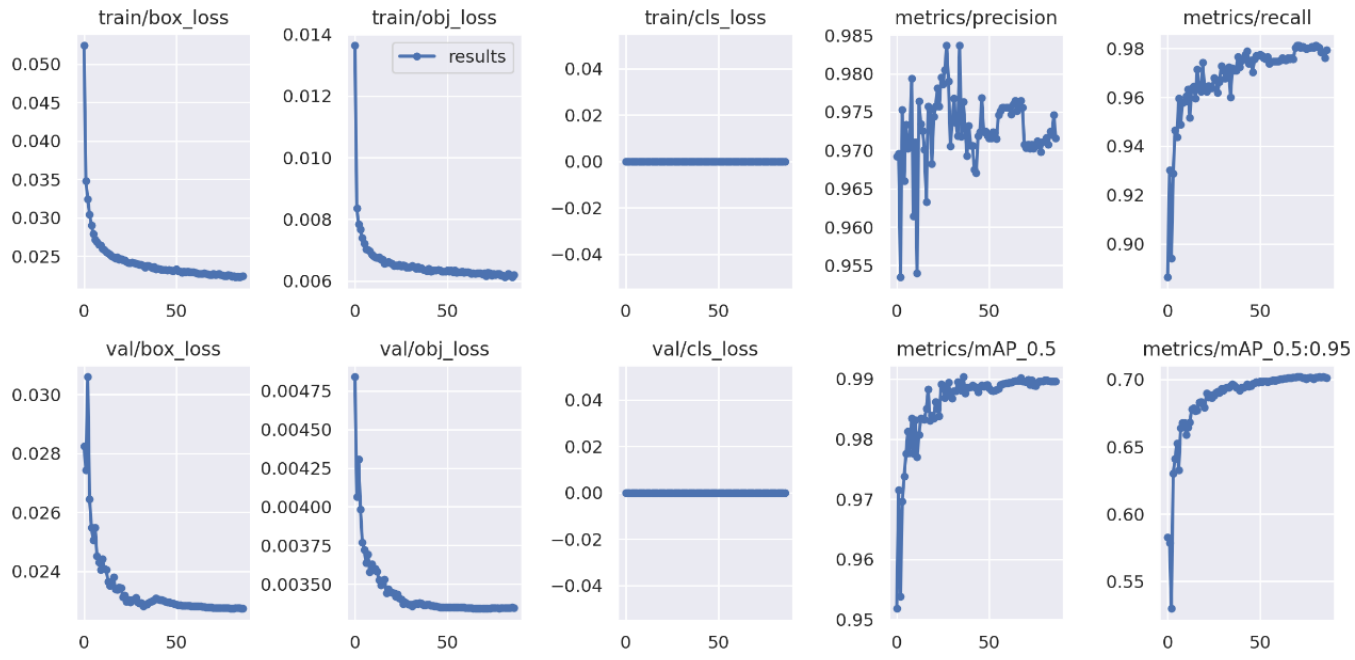
We used a total of 24,242 images for training, with the following distribution:

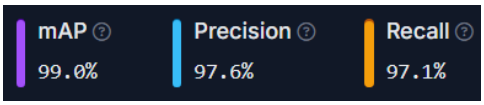


The data set can be accessed at the following link: License Plate Recognition Dataset.

Model Training and Evaluation

The following graphs represent the training progress and final Evaluation of the model:





2 YOLOv8n.pt

This is a pre-trained model trained on the COCO dataset. It can be downloaded from: YOLOv8n.pt on Hugging Face.

Although the YOLOv8n.pt model has the capability to classify the following classes, we will only focus on detecting the classes with number plates:

person	bicycle	car	motorcycle	airplane
bus	train	truck	boat	traffic light
fire hydrant	stop sign	parking meter	bench	bird
cat	dog	horse	sheep	cow
elephant	bear	zebra	giraffe	backpack
umbrella	handbag	tie	suitcase	frisbee
skis	snowboard	sports ball	kite	baseball bat
baseball glove	skateboard	surfboard	tennis racket	bottle
wine glass	cup	fork	knife	spoon
bowl	banana	apple	sandwich	orange
broccoli	carrot	hot dog	pizza	donut
cake	chair	couch	potted plant	bed
dining table	toilet	tv	laptop	mouse
remote	keyboard	cell phone	microwave	oven
toaster	sink	refrigerator	book	clock
vase	scissors	teddy bear	hair drier	toothbrush

3 main.py

```
# Load models
coco_model = YOLO('yolov8n.pt')
license_plate_detector = YOLO('license_plate_detector.pt')
```

Initialize two YOLO object detection models:

- `coco_model`: This is a pre-trained YOLO model loaded from the `yolov8n.pt`.
- `license_plate_detector`: This is a specialized YOLO model for detecting license plates, loaded from the `license_plate_detector.pt`.

```
# Load video
cap = cv2.VideoCapture('sample.mp4')
```

- `'sample.mp4'`: The path to the video file to be loaded.

```
ret, frame = cap.read()
```

- `ret`: a boolean value that indicates whether the frame was read successfully.
- `frame`: The actual frame of the video captured as a NumPy array.

```
detections = coco_model(frame)[0]
detections_ = []
for detection in detections.boxes.data.tolist():
    x1, y1, x2, y2, score, class_id = detection
    if int(class_id) in vehicles:
        detections_.append([x1, y1, x2, y2, score])
```

- The model (`coco_model`) processes the input frame and returns all detected objects.
- `for detection in detections.boxes.data.tolist():` Iterates over all detected objects. Each `detection` contains the bounding box coordinates, score, and class ID.
- `x1, y1, x2, y2, score, class_id = detection`: Extracts individual components of a detection:
- `if int(class_id) in vehicles:` Filters detections by checking if the `class_id` corresponds to a vehicle class (defined in a separate list, `vehicles`).

```
[2443.468017578125, 592.7649536132812, 2614.2138671875, 763.5208740234375, 0.366671085357666, 2.0]
[843.773681640625, 578.8257446289062, 969.6482543945312, 691.3671264648438, 0.35869085788726807, 2.0]
[1316.564697265625, 613.21826171875, 1443.6754150390625, 704.7024536132812, 0.3052750825881958, 2.0]
```

```
# Track vehicles
track_ids = mot_tracker.update(np.asarray(detections_))
```

- SORT : A simple online and realtime tracking algorithm for 2D multiple object tracking in video sequences.
- Assign and maintain unique IDs for each vehicle in video frames.
- Continuously track vehicle movements in the scene, even if some frames have missed detections.

```
# Assign license plate to car
xcar1, ycar1, xcar2, ycar2, car_id = get_car(license_plate, track_ids)
```

- This step ensures that the detected license plate is associated with the correct car based on the vehicle's position and unique tracking ID. By combining information from the license plate detection model and the MOT tracker, the system can match the license plates with the cars they belong to.

get_car from utils.py

```
foundIt = False
for j in range(len(vehicle_track_ids)):
    xcar1, ycar1, xcar2, ycar2, car_id = vehicle_track_ids[j]

    if x1 > xcar1 and y1 > ycar1 and x2 < xcar2 and y2 < ycar2:
        car_indx = j
        foundIt = True
        break

    if foundIt:
        return vehicle_track_ids[car_indx]
```

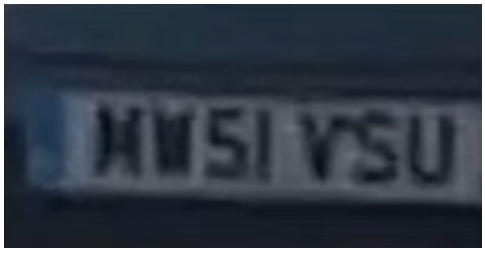
For each vehicle's bounding box, it checks if the coordinates `(x1, y1, x2, y2)` of a detected license plate are completely within the bounding box of the vehicle.

```
# Crop license plate
license_plate_crop = frame[int(y1):int(y2), int(x1): int(x2), :]

# Process license plate
license_plate_crop_gray = cv2.cvtColor(license_plate_crop, cv2.COLOR_BGR2GRAY)
_, license_plate_crop_thresh =
    cv2.threshold(license_plate_crop_gray, 64, 255, cv2.THRESH_BINARY_INV)

# Read license plate number
license_plate_text, license_plate_text_score = read_license_plate(license_plate_crop_thresh)
```

- processes the detected license plate region by cropping it out, converting it to grayscale, applying thresholding to enhance the features, and then reading the text (license plate number) from the image.



original cropped plate

Binary Thresholded

4 test.csv and test_interpolated.csv

As a result of `main.py`, the output is written into `test.csv`, which contains the following columns:

Column Name	Description
`frame_nmr`	Frame number of the video.
`car_id`	Unique identifier assigned to each car.
`car_bbox`	Coordinates of the car's bounding box (x1, y1, x2, y2).
`license_plate_bbox`	Coordinates of the license plate's bounding box (x1, y1, x2, y2).
`license_plate_bbox_score`	Confidence score for the detected license plate bounding box.
`license_number`	Recognized license plate number as text.
`license_number_score`	Confidence score for the recognized license plate number.

frame_nm	car_id	car_bbox	license_plate_bbox	license_plate_bbox_score	license_number	license_number_score
0	3	[752.8723754882812 1369.	[973.7656860351562 1753.3	0.5657119154930115	OA13NRU	0.31111454698547064
1	5	[2197.3992555005066 115	[2407.91455078125 1551.37	0.5950816869735718	MU51KSU	0.12425747192016284
1	3	[752.7640049748125 1369.	[973.7571411132812 1753.3	0.5656632781028748	OA13NRU	0.18627882204947346
2	5	[2201.178826418825 1163.	[2411.8359375 1549.122802	0.5861984491348267	HV51VSU	0.20408029170235567
4	3	[749.085051577626 1380.2	[963.6485595703125 1783.3	0.4999430179595947	NA13NRU	0.46606421649012747
4	5	[2196.699689616603 1171.	[2410.735107421875 1562.9	0.4264005124568939	HU51KSU	0.2282599848341539
5	5	[2194.09660696655 1174.1	[2411.2822265625 1562.439	0.4976135492324829	HU51VSU	0.4106486276346287

5 Addressing Ambiguities in CSV Data and OCR Results

Issues Identified

- **Ambiguity in Car IDs Across Frames:** For a specific car ID, its bounding box and license plate information are not consistently detected in all frames. This results in ambiguity when tracking vehicles, as cars appear and disappear in the video playback.
- **Variability in OCR Results:** The OCR process may yield multiple different license plate readings for the same car. Issues like extra spaces, special characters, or misread letters can introduce inconsistencies in the recognized license numbers.

Proposed Solutions

1. **Linear Interpolation for Missing Frames:** To address the appearance and disappearance of car IDs in frames, we perform linear interpolation of the bounding boxes for a specific car ID. This ensures that bounding boxes are consistently present for a car across its lifespan in the video.
2. **Selecting the Best OCR Result:** After interpolating the bounding boxes, the license number for each car is determined by selecting the OCR result with the highest confidence score across all frames. This ensures that:
    - Inconsistent OCR results are replaced with the most reliable reading.
    - A continuous and consistent license plate number is assigned to each car.

Benefits of the Approach

- Eliminates the issue of cars flickering in and out of detection during video playback, providing a smoother tracking experience.
- Improves the accuracy and reliability of license plate recognition by using the best OCR results.
- Ensures consistency in tracking car IDs, even when detection is imperfect or partial in certain frames.

using test_interpolated.csv in visualize.py to select best OCR score

```
license_plate = {}
for car_id in np.unique(results['car_id']):
    # Extract the max score for the license number
    max_score =
        np.amax(results[results['car_id'] == car_id]['license_number_score'])

    # Retrieve license plate number and corresponding frame number
    matching_row = results[(results['car_id'] == car_id) &
                           (results['license_number_score'] == max_score)]
    license_plate[car_id] = {
        'license_crop': None,
        'license_plate_number': matching_row['license_number'].iloc[0]
    }
```

6 Results