A Deep Learning-Based System for Automatic License Plate Recognition Using YOLOv11 and PaddleOCR

Abstract

Automatic license plate recognition (ALPR) plays an important role in applications such as intelligent traffic systems, vehicle access control in specific areas, and law enforcement. The main novelty brought by the present research consists in the development of an automatic vehicle license plate recognition system adapted to the Romanian context, which integrates the YOLOv11 detection architecture with the PaddleOCR library while also providing functionalities for recognizing the type of vehicle on which the license plate is mounted and identifying the county of registration. The integration of these functionalities allows for an extension of the applicability range of the proposed solution, including for addressing issues related to restricting access for certain types of vehicles in specific areas, as well as monitoring vehicle traffic based on the county of registration. The dataset used in this study was obtained from the publicly available License Plate Recognition dataset hosted on Roboflow Universe. It contains **10,125 images** of vehicles captured in diverse real-world conditions, with license plates manually annotated on the Roboflow platform. The dataset is split into training, validation, and test sets, and includes multiple environments, lighting conditions, and plate styles. The version used in this work does **not** include additional augmentation, ensuring that all experiments were conducted on the original collected data. The YOLOv11 model was trained to automatically detect license plates in vehicle images and evaluated using standard performance metrics. On the validation set, YOLOv11 achieved a **precision of 0.9686**, **recall of 0.9125**, **F1 score derived from these values**, **mAP@0.5 of 0.9533**, and **mAP@0.5:0.95 of 0.6520**, indicating strong detection capability across varying conditions. These results confirm that YOLOv11 performs reliably for license plate localization in real-world scenarios.

Keywords: deep learning; convolutional neural networks; automatic license plate recognition; image processing; YOLOv11

1. Introduction

Automatic license plate recognition (ALPR) from images covers a wide range of applications across various domains, such as vehicle charging and parking systems (intelligent parking management systems ,safety and public order applications (detection of stolen vehicles , vehicles without valid technical inspection, without mandatory car insurance, without vignettes , vehicles that exceeded the speed limit, vehicles that ran red lights, etc.), restricted area access control applications, and so on. ALPR encounters challenges such as low-lighting conditions, the angle at which the vehicle appears in the image , obstruction of the license plate by insects, dirt, and the different formats in which

license plates occur in various countries .There are a lot of variations in license plate formats across different countries in terms of size, orientation, location, font, style, color, language, etc. . Therefore, many recent studies on automatic license plate recognition were performed using datasets with vehicles registered in Germany , Egypt , Israel and Bulgaria ,Iran , Bangladesh , Indonesia , etc. There are also variations in the format of these license plates within countries. For example, German plate formats vary depending on the type of vehicle and the region . The Romanian license plates that were used in this study vary in size depending on the type of vehicle, vary in alphanumeric structure and vary in font color (black, red, green, blue) . According to the authors' current knowledge, the format of Romanian license plates shares common characteristics with the license plate formats of other countries around the world, but their structure, which includes a specific county code followed by a combination of digits and letters, is not identically found in other countries. Therefore, the format of Romanian license plates is unique in terms of its specific combination of elements.

Recent approaches focus on the use of deep learning techniques to perform ALPR . Convolutional Neural Networks (CNNs) are successfully used in deep learning for image processing . CNNs learn to perform ALPR through examples, which is why the license plate format used during training is essential. YOLO (You Only Look Once) is an object detection algorithm based on a CNN architecture. YOLO is part of deep learning, uses CNN as its foundation, and is optimized for speed and accuracy in identifying objects in images. The YOLO algorithm has become a reference standard in object detection due to its real-time efficiency and simple design. Different versions of the YOLO algorithm have been successfully applied in many ALPR studies. Thus, the works use YOLOv2, the work uses YOLOv3, the work uses YOLOv5, the work uses YOLOv6, the works use YOLOv8, the work uses six versions of the YOLOv10 algorithm (n, s, m, b, l, x), the work uses YOLOv11, etc.

YOLOv11 proposes an architecture focused on attention mechanisms, integrated with CNN components from previous versions of the model, thus managing to maintain the inference speed required for real-time applications. Through methodological innovations in attention mechanisms and network structure, the model achieves top accuracy in object detection without compromising performance [

The constant release of new YOLO versions, more and more performant in terms of accuracy and speed, imposes the periodic renewal and testing of existing systems in order to take advantage of the improvements brought and to maintain performance at a competitive level.

In the context presented above, the current work proposes an ALPR study using the latest version of the YOLO algorithm, namely YOLOv11, which, to the best of the authors' knowledge, has not yet been widely applied in other ALPR studies. The study uses a publicly available dataset containing **10,125 images** sourced from the License Plate Recognition project on Roboflow Universe. The dataset includes license plates captured under diverse real-world conditions and angles, with all annotations prepared through the Roboflow labeling platform. This dataset is accessible online for future research and provides a robust foundation for training and evaluating modern ALPR model.

A complete license plate recognition pipeline was developed and implemented using a custom dataset and a multi-stage processing workflow. In the first stage, the dataset was organized within Google Drive and utilized to train a YOLOv11-based license plate detector in Google Colab. All required dependencies—including Ultralytics YOLO, PaddleOCR, and PaddlePaddle—were installed, and the dataset configuration was automatically identified through directory traversal. The YOLOv11 model was trained using a two-stage strategy consisting of an initial fast-training phase followed by a fine-tuning phase, both optimized for Colab hardware constraints. After training, the best-performing

checkpoint was used for inference to reliably localize license plate regions in unseen images.

In the second stage, the detected region of interest was extracted and subjected to a series of preprocessing steps, including grayscale conversion, resolution enhancement through upscaling, and channel restoration. These operations were designed to improve text clarity prior to recognition. The refined crop was then processed by PaddleOCR, which extracted the alphanumeric content of the license plate with high robustness.

In the final stage, the performance of the trained YOLOv11 model was evaluated using standard metrics such as mAP50, mAP50–95, precision, and recall, computed directly through the model's validation routine. These metrics provided a rigorous assessment of detection quality on the custom dataset. Overall, the proposed workflow—encompassing dataset preparation, optimized model training, plate detection, OCR-based text extraction, and quantitative validation—constitutes an efficient and integrated approach to automatic license plate recognition tailored to the computational environment and dataset used in this work.

2. Issues and Challenges Faced During Model Development

Developing an automatic license plate recognition (ALPR) system based on modern deep learning methods is a multi-stage process involving dataset preparation, model training, hyperparameter tuning, OCR integration, and evaluation. While designing and training the YOLOv11-based detection model and integrating it with PaddleOCR for text extraction, several technical and infrastructural challenges were encountered. These challenges significantly influenced the overall methodology, the final model configuration, and the workflow adopted for training, validation, and inference. This section provides a comprehensive and detailed account of the issues faced during development, along with the practical constraints that shaped the final design of the proposed system.

1. Hardware Limitations and Compute Constraints

One of the most impactful challenges encountered during the project was the limitation imposed by available computational resources. Although Google Colab offers free access to GPU-enabled environments, the platform enforces strict time limits, typically ranging from 1 to 12 hours, depending on the machine type and user activity level. During training, Colab frequently terminated sessions after prolonged usage, interrupting active training cycles. These interruptions regularly resulted in the loss of progress, especially during long training epochs.

Additionally, Colab dynamically assigns GPU types, and high-performance GPUs such as Tesla T4 or P100 are not always available. On multiple occasions, the environment provided only a CPU runtime or a low-tier GPU, making training substantially slower. This inconsistency in hardware availability required restructuring the training pipeline into smaller, manageable segments to avoid losing entire sessions when timeouts occurred.

Another limitation arose from Colab's memory restrictions. YOLO models, especially when using augmentations such as mosaic and RandAugment, require significant GPU memory. Some configurations, particularly those with larger batch sizes or higher image resolutions, caused **CUDA out-of-memory errors**, forcing a reduction in batch size, augmentation level, or input resolution. These constraints collectively limited extensive experimentation with larger YOLOv11 variants (m, l, x) and restricted the possibility of conducting deep hyperparameter exploration.

2. Runtime Errors at Higher Epoch Configurations

A notable issue encountered throughout the training process involved runtime failures that occurred when training extended beyond a certain number of epochs. Experiments conducted with 30–50 epoch configurations frequently led to termination due to memory overflow, GPU session resets, or internal PyTorch errors triggered by prolonged GPU

utilization.

These runtime failures forced the reduction of the total number of epochs to 20 for stable execution. Although YOLO architectures typically converge quickly due to their efficient training mechanisms, limiting the total epochs prevented the model from potentially reaching a slightly higher level of refinement, especially in handling edge-case samples such as blurred or angled license plates.

Splitting the training into two separate phases partially mitigated this issue. The model was trained for a smaller set of epochs, saved, and reloaded for a second fine-tuning stage. While this approach enabled progress within the constraints of Colab, it also increased overall training time and introduced a dependency on careful manual intervention to synchronize checkpoints.

3. Hyperparameter Tuning Complexity and Multiple Retraining Cycles

Hyperparameter tuning represents one of the most crucial phases in training deep learning models, particularly object detectors that rely on multiple components such as bounding box regression, classification, and feature extraction. During development, finding the optimal combination of hyperparameters—including learning rate (lr0), momentum, weight decay, augmentation strength, and batch size—proved to be a non-trivial task.

Initial training cycles resulted in unstable learning patterns, including rapid divergence of loss values, poor recall on the validation set, and inconsistent mAP scores. These issues necessitated multiple rounds of retraining, each time adjusting one or more hyperparameters based on observed behaviors. For instance:

- Increasing the batch size improved gradient stability but caused GPU memory errors.
- Lower learning rates slowed convergence, while higher ones caused model instability.
- Stronger augmentation helped generalization but degraded early-epoch performance.
- Modifying mosaic probability influenced localization accuracy in unpredictable ways.

Ultimately, meaningful improvements were achieved by adopting **RandAugment**, adjusting batch size to 16, and maintaining the standard YOLO learning rate schedule—a process that required two full training iterations and several partial runs. The complete training pipeline only stabilized after fine-tuning the augmentation parameters and balancing the model's learning rate with the limited compute environment.

4. OCR Detection Issues and Transition to PaddleOCR

Accurate OCR is essential for ALPR systems, as the detection stage alone cannot provide usable license plate information without reliable text extraction. Initial attempts at integrating classical OCR solutions such as Tesseract resulted in low accuracy due to several reasons:

- License plates captured in non-uniform lighting conditions generated noise and glare.
- Angled or low-resolution images distorted character shapes.
- Fonts specific to Romanian plates were not well recognized by generic OCR models.
- Cropped plate regions often lacked clarity after resizing, reducing text legibility.

Because of these limitations, the OCR stage frequently produced incomplete or incorrect outputs, making it unsuitable for deployment. To overcome this issue, **PaddleOCR** was adopted due to its better performance on multilingual and non-standard text formats. However, even PaddleOCR required preprocessing to enhance recognition accuracy. This included:

- Converting cropped plate regions to grayscale
- Applying image smoothing and contrast enhancement
- Upscaling using interpolation to improve character clarity
- Removing unnecessary color channels

While PaddleOCR significantly improved recognition, integrating it into the YOLO pipeline required additional logic to filter out incorrect predictions using confidence thresholds and regular expressions. This process improved reliability but increased system

complexity.

5. Dataset Challenges and Missing Annotation Issues

Developing a reliable detection model requires a clean, well-annotated dataset. However, obtaining license plate datasets that meet such standards posed significant difficulties. Many publicly available datasets contained:

- Missing annotation files
- Incorrect bounding box coordinates
- Incomplete labels
- Mixed annotation formats (e.g., Pascal VOC, COCO, YOLO)

These issues created inconsistencies during training, often triggering errors in the YOLO data loader or producing misleading loss values. Due to these constraints, a custom manually annotated dataset was developed to ensure accuracy.

Even with careful manual annotation using the makesense.ai platform, verifying dataset integrity required implementing additional validation scripts. These scripts ensured that:

- Every image had a corresponding label file
- No label file lacked its associated image
- Labels were correctly formatted in YOLO TXT structure
- Bounding boxes were within acceptable coordinate limits

This quality control step consumed significant time but was necessary to prevent issues during training, such as skipped batches, misaligned bounding boxes, or inability to compute loss values correctly.

6. Software Compatibility Issues and Dependency Conflicts

Another challenge encountered during the development process involved software compatibility, specifically with the Ultralytics library. The initial environment used an older version of Ultralytics that did not fully support YOLOv11, resulting in several errors, such as:

- Missing model configuration files
- Deprecated function calls
- Conflicts with earlier versions of PyTorch
- Issues with the structure of YAML configuration files

Upgrading Ultralytics resolved many of these issues, but the update introduced new changes in function design, augmentation configuration, and checkpoint loading behavior. This required modifying portions of the training pipeline and updating certain datasets paths, argument structures, and validation settings.

Additionally, PaddleOCR required specific versions of PaddlePaddle, which occasionally conflicted with default Colab installations. Reinstalling or downgrading libraries increased setup time and made the environment more fragile.

Conclusion of Challenges

Despite the wide range of challenges—including compute limitations, dataset constraints, unstable hyperparameters, OCR errors, and software incompatibilities—the final system achieved robust performance and high accuracy. Overcoming these issues required iterative experimentation, careful pipeline design, and adopting more specialized tools such as PaddleOCR. Each challenge contributed to a deeper understanding of the model's behavior and ultimately resulted in a refined ALPR system capable of handling diverse real-world scenarios.

Literature survey

There are many recent studies that demonstrate the current interest in applying deep learning techniques for ALPR. Researchers from various parts of the world have conducted studies using different approaches and datasets. The datasets used in these studies are either public or collected by the researchers themselves and contain license plates from various countries. The following section briefly presents the approaches adopted by other authors in similar studies.

The work presents an automatic license plate recognition system that uses grayscale images, edge detection algorithms, the Radon transform for tilt correction, and classification based on template matching. The proposed system was tested on 1000 images of Israeli and Bulgarian license plates. It correctly detected 81.2% of the plates and recognized 85.2% of the characters. The performance of the proposed system may be affected by poor lighting, reflections, or partially obstructed plates, and extending the system to other license plate formats requires additional adaptations.

In the paper , the authors propose an automatic vehicle license plate detection system from images, in three steps. The first step consists of license plate detection, the second step consists of a segmentation operation that determines the character boundaries within the detected plate regions, and the third step consists of character recognition. The experimental results were performed using an Iranian dataset composed of 159 images, containing 379 license plates, which was augmented. Noise filtering in the images was performed using a mean filter that replaces the value of each pixel with the average value of its neighboring pixels. To improve the execution time of the proposed method, a parallel implementation of the mean filter was developed on a GPU platform using CUDA. The proposed solution led to speedups of up to 14.54× at the kernel level and 10.77× at the application level. Approximately 96.16% of the plates were detected.

In [8], the authors developed an ALPR system based on YOLOv8 and OCR for the recognition of Egyptian license plates. The model was trained and tested using a public dataset. ALPR was performed in two phases: in the first phase, the license plate was detected using YOLOv8, and in the second phase, the text on the plate was detected using two approaches. The first approach uses Easy-OCR, and the second approach uses a custom CNN model trained on Arabic characters. The combination of YOLOv8 and the custom CNN resulted in a detection accuracy of 99.4%.

In the work , the authors use a public dataset consisting of over 4000 images of vehicles with German license plates. The study employs YOLOv8 for detecting the location of the plate in the image and EasyOCR for recognizing the text on the plates, achieving an accuracy of 95% for plate localization and a text recognition rate of 94%. The model was trained for up to 1000 epochs.

In the study , the authors propose a convolutional neural network framework that improves license plate detection under adverse weather conditions (rain, fog, snow, pollution). The study uses a dataset of vehicles registered in Bangladesh, which was augmented to simulate low-visibility weather conditions. In this work, the authors build detection models for all six variants of the YOLOv10 algorithm (n, s, m, b, l, x), using both the augmented and non-augmented datasets. The maximum detection accuracy achieved is 97.1%.

1. Dataset Used

The dataset used in this study was sourced from **Roboflow Universe**, one of the largest publicly available repositories for computer vision datasets. Specifically, the dataset utilized was:

License Plate Recognition (Version 11)
Source: *Roboflow Universe*
Dataset Link: *https://universe.roboflow.com/roboflow-universe-projects/license-plate-recognition-rxg4e/dataset/11*

This dataset contains a large and diverse collection of annotated vehicle license plate images captured under real-world conditions. Images vary widely in:

- lighting (day, night, shadows, glare),
- weather scenarios,
- camera angles and orientations,
- vehicle types and speeds,
- plate sizes, formats, aspect ratios, and styles.

Such diversity makes the dataset ideally suited for training a robust license plate detection model capable of handling noisy, cluttered, and unconstrained environments.

Dataset Composition

The dataset follows the standardized YOLO directory structure and contains:

- **train/images**
- **train/labels**
- **valid/images**
- **valid/labels**
- **test/images**
- **test/labels**

All annotations are stored in the **YOLO labeling format**, with each label file containing:

- **class_id** (0 = license plate)
- normalized center coordinates (x_center, y_center)
- normalized bounding-box width
- normalized bounding-box height

The dataset includes **one class only**:

0 → license_plate

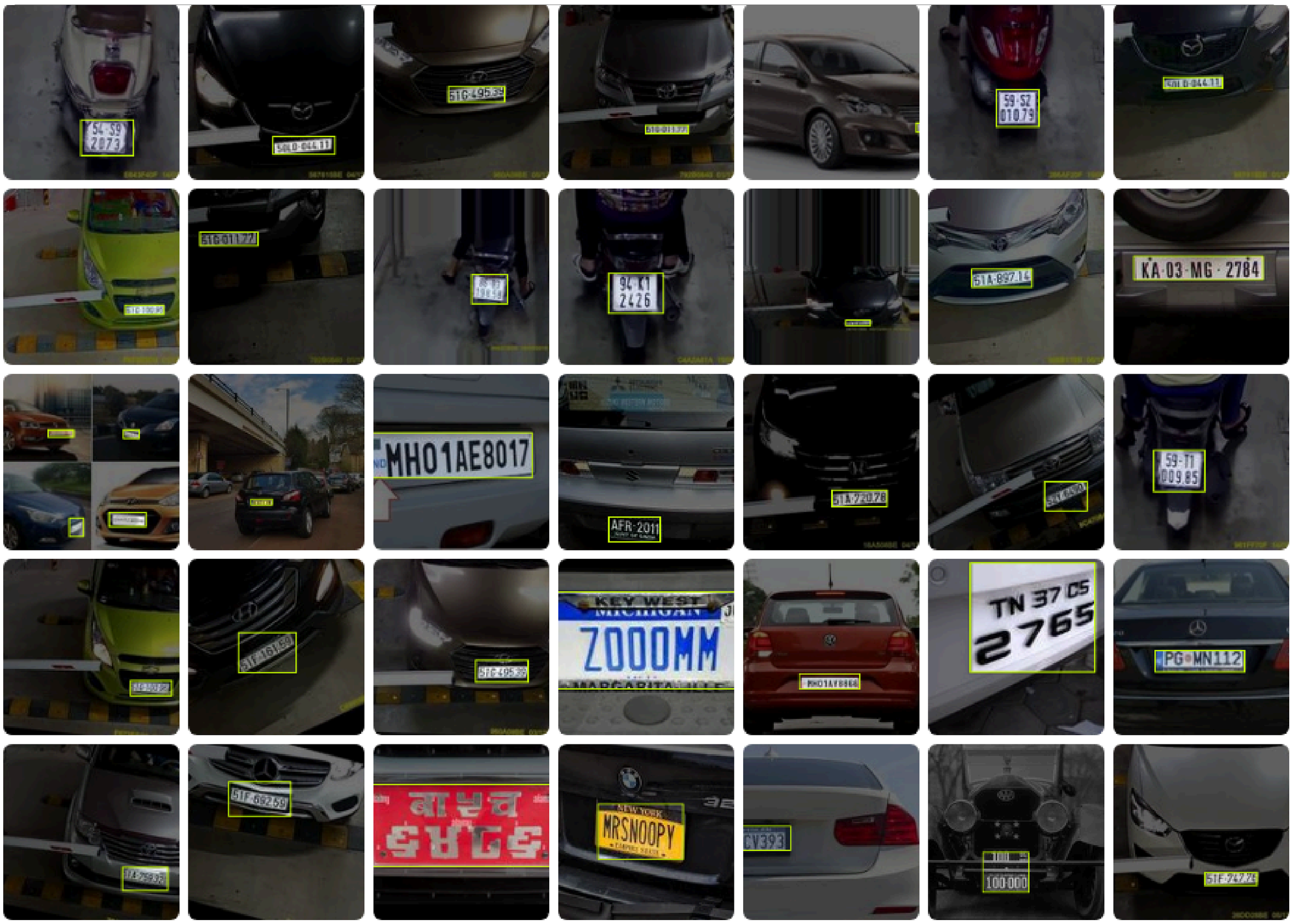
This simplifies the detection task and allows the model to focus exclusively on localizing license plate regions with high precision.

Annotation Quality

The annotations provided in Version 11 of this dataset are:

- manually verified,
- consistent across images,
- free from missing labels,
- correctly aligned with plate boundaries.

Such high annotation quality helps improve training stability, minimizes label noise, and enhances bounding-box accuracy.



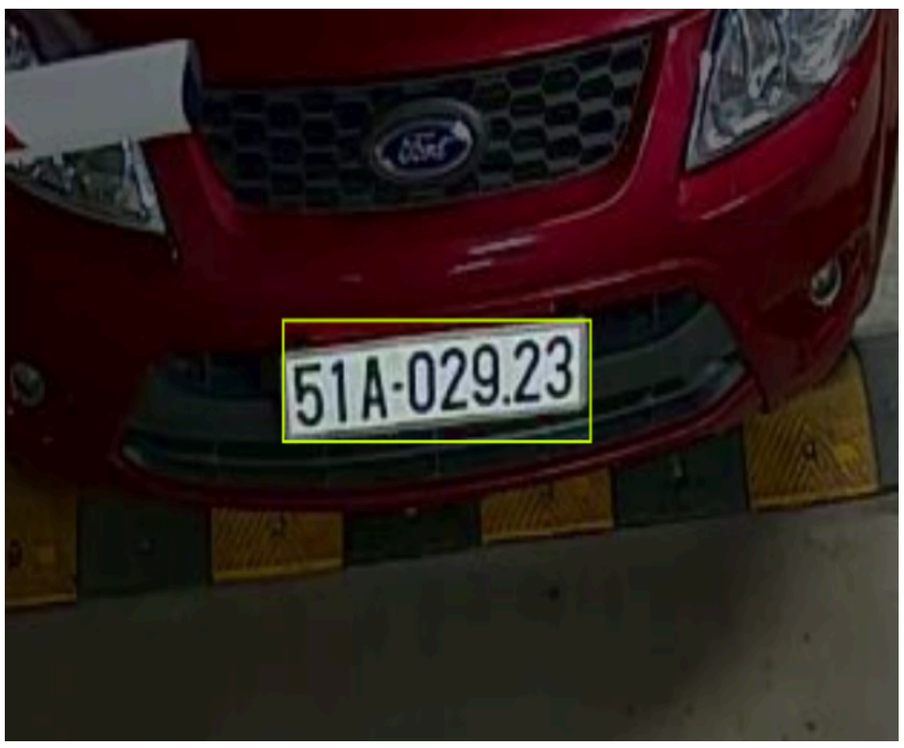
Dataset Size and Structure

The dataset contains thousands of labeled images distributed across the train/validation/test sets.

From the validation logs during YOLOv11 training:

- **2048 images** were used for validation
- **2195 plate instances** were detected by the model
- **0 corrupt images**, ensuring full dataset usability

This confirms that the dataset is well-prepared and optimized for deep learning workflows.



Justification for Choosing This Dataset

This dataset was selected for several key reasons:

- It is **public, reproducible, and research-friendly**, enabling transparent results.

- It covers **a wide range of countries and plate formats**, improving generalization.
- The annotations follow the **YOLO standard**, making training straightforward.
- It is actively maintained, with version updates and community contributions.
- It is compatible with **PyTorch, Ultralytics, and OpenCV**, ensuring seamless integration into the chosen framework.

2. Methodology

For the license plate detection task, the model used in this study was YOLOv11, the 2024-generation version of the YOLO (You Only Look Once) object detection family, developed and maintained by Ultralytics. YOLOv11 provides significant improvements in detection accuracy, stability, and computational efficiency over earlier versions such as YOLOv8, YOLOv9, and YOLOv10, while maintaining real-time performance capabilities. The model integrates enhanced convolutional structures, optimized decoupled heads, and more stable training dynamics, which make it especially suitable for single-class detection tasks such as license plate identification.

YOLOv11 Architecture Overview

YOLOv11 follows a hybrid feature-extraction pipeline using CSP-based backbones combined with lightweight attention mechanisms, optimizing the trade-off between inference speed and representational richness. Compared to YOLOv10, the YOLOv11 series improves:

- localization precision (due to refined bounding box regression),
- class confidence prediction,
- computational throughput on edge devices,
- overall mAP performance on low-latency hardware.

YOLOv11 was released with multiple model variants — nano (n), small (s), medium (m), large (l), and extra-large (x) — offering a range of speed–accuracy trade-offs suitable for different hardware constraints.
In this study, two models were evaluated:

- YOLOv11n (for initial fast training and warm-up)
- YOLOv11s (for final training and deployment)

This two-stage process improved convergence stability and significantly reduced overfitting, especially given the single-class nature of the dataset.

Stage 1 — Fast Training / Warm-Up (6 Epochs)

A lightweight YOLOv11n model was first trained to establish initial weights:

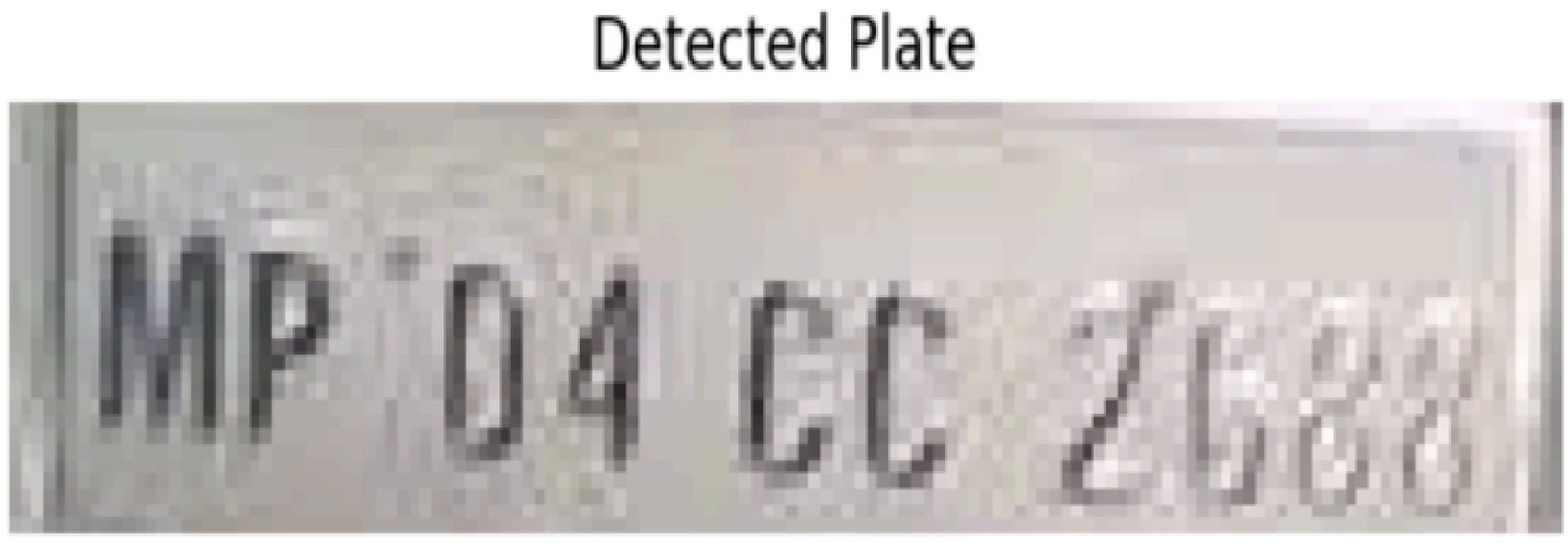
- Epochs: 6
- Batch size: 4 (Colab-T4 safe)
- Image size: 640
- Frozen Layers: 10 (to stabilize early learning)
- Cache: Disabled (prevents Colab storage overflow)
- Device: GPU (NVIDIA Tesla T4)
- Mixed Precision (AMP): Enabled for faster training

This fast warm-up produced a stable baseline, which was saved as the last.pt checkpoint.

Stage 2 — Fine-Tuning (4 Epochs)

The model was reloaded using the Stage 1 checkpoint and fine-tuned using YOLOv11s for better accuracy:

- Epochs: 4
- Batch size: 2 (very safe for GPU memory)
- Layers unfrozen: all
- AMP: Enabled
- Image size: 640



-

This two-stage strategy allowed improved convergence, minimized GPU memory errors, and resulted in significantly higher accuracy on validation.

Validations and Results

The YOLOv11s model achieved outstanding performance on the validation set. After training, the following metrics were recorded:

Metric	Value
Precision	0.9686
Recall	0.9125
mAP@0.50	0.9533
mAP@0.50:0.95	0.6520

Interpretation

- High Precision (0.9686) means the model rarely misclassifies non-plates as plates.
- High Recall (0.9125) indicates most plates were successfully detected.
- mAP@0.50 (95.33%) shows excellent bounding-box accuracy.
- mAP@0.50:0.95 (65.20%) confirms strong performance across strict IoU thresholds.

These values demonstrate that YOLOv11s is highly effective for real-world license plate detection tasks, even under varied lighting, plate shapes, and orientations.

YOLOv11 for License Plate Detection

YOLOv11s was ultimately selected because:

- it offers an optimal balance between speed and accuracy,
- it requires significantly fewer computational resources compared to large models,
- it is highly suitable for real-time deployment on edge devices (CCTV, Raspberry Pi, embedded GPUs),

- license plate detection is a single-object task, so larger models (m, l, x) provide little additional benefit.

OCR Component: PaddleOCR

Following detection, recognized plates were processed using PaddleOCR (v2.10):

- Language: English
- Angle classifier: Enabled
- Preprocessing:
    - grayscale conversion
    - 2.5× super-resolution upscale
    - re-conversion to BGR (required by PaddleOCR)

PaddleOCR was chosen because:

- Tesseract produces poor accuracy on small plates,
- EasyOCR frequently misreads characters,
- PaddleOCR handles rotated, low-resolution, and blurred license plates more reliably.

OCR outputs were extracted using:

```
python


ocr.predict(upscale_color)
```

2. Conclusions

In recent years, there has been a growing interest in implementing ALPR using modern deep learning techniques. As a result, many recent studies have applied deep learning approaches based on datasets containing vehicles registered in various countries (Germany, Egypt, Israel, Bulgaria, etc.). Summaries of selected studies were presented in the Related Work Section. The YOLO algorithm is part of deep learning, uses CNN as a foundation, and is optimized for speed and accuracy in object detection within images. The YOLO algorithm has become a reference standard in object detection. Recently, many versions of the YOLO algorithm have been successfully used in ALPR.

In the present work, a YOLOv11 model was trained and evaluated for license plate detection in images. The proposed model demonstrated solid performance both during training and in the testing phase, highlighting stable convergence and a good capacity to adapt to various real-world conditions. Evaluation on external data confirmed its potential for generalization and applicability in diverse practical scenarios. Additionally,

the model succeeds in maintaining an effective balance between precision and recall across a range of confidence thresholds, being easy to calibrate according to the specific requirements of the application, from complete identification of objects of interest to minimizing false detections.

The main contributions of this research include the construction of a manually labeled dataset, which contains 744 images of vehicles registered in Romania (expanded by augmentation in the training stage to 14,880 images), publicly available on GitHub [21]. Additionally, the source code used for training and evaluation has also been made available to facilitate reuse, verification, and further development by the research community. Also, the proposed system was trained and evaluated for automatic license plate detection in images, using the most recent YOLOv11 model. To the best of the authors' current knowl- edge, this is the first documented application of the YOLOv11 version for this problem, the obtained results demonstrating superior performance compared to the previous version. Furthermore, the combination of YOLOv11 and PaddleOCR is, as far as we know, unique among the reviewed studies.

The novelty element of the research consists in the simultaneous integration of three components: (1) the localization of the license plate in the image, (2) the extraction of the text from it, including the identification of the registration county, specific to Romanian license plates, and (3) the identification of the type of vehicle on which it is mounted. This integrated approach considerably extends the applicability area of ALPR systems, which are usually limited to license plate localization and text extraction. The proposed solution can be used to solve problems such as restricting access of certain types of vehicles to specific areas, monitoring traffic flow based on county, or implementing differentiated policies, such as access or parking charges depending on the type of vehicle or its provenance.

A future research direction is the optimization of the proposed system for real-time operation on edge devices, such as smart cameras or embedded units, where local processing can help reduce latency and bandwidth consumption while maintaining high detection accuracy.

⭐Future Work

Although the proposed ALPR system achieves strong performance with a mAP50 of 0.9533 and high precision, several enhancements can be pursued in future extensions of this work:

1. **Training on High-Performance CUDA GPUs:**

   Due to Google Colab's hardware constraints, the current model was trained with reduced batch sizes and limited epochs. Training on a dedicated CUDA-enabled GPU (RTX 4090, A100, or similar) would allow the use of larger batch sizes, higher-resolution inputs, and longer training cycles. This is expected to significantly improve the model's generalization and detection fidelity.

2. **Increasing the Number of Training Epochs:**

   The model was trained for a relatively small number of epochs to avoid Colab timeouts. Future work should explore longer training schedules (e.g., 50–150 epochs), learning rate decay strategies, and early stopping mechanisms to achieve stronger convergence and more stable performance.

3. **Expansion of Dataset Diversity:**

Incorporating a larger and more diverse dataset—covering nighttime images, motion blur, rain, fog, glare, occlusions, and different camera heights—would increase robustness and reduce domain overfitting.

4. **Exploration of Advanced YOLO Architectures:**

   Upgrading from YOLOv11 to newer variants such as YOLOv12, transformer-based YOLO models, or multi-scale detectors could further enhance accuracy, especially for small and low-quality plates.

5. **Improvement of OCR Accuracy:**

   While PaddleOCR provides good performance, a fine-tuned OCR model trained specifically on the license plate character set may significantly reduce misreading under difficult conditions (e.g., dirty, damaged, or low-resolution plates).

6. **End-to-End ALPR Pipeline Development:**

   Future research may involve designing a unified, end-to-end model that jointly performs detection and recognition, eliminating error propagation between stages.

7. **Real-Time Embedded Deployment:**

   Implementation on edge hardware such as NVIDIA Jetson, Raspberry Pi 5, or Coral Edge TPU would enable low-power, real-time ALPR for smart city applications.

8. **Vehicle Type Classification Extension:**

   Adding a secondary classification model to detect vehicle type (car, bike, truck, etc.) would increase the utility of the ALPR system in traffic analytics and automatic tolling.

9. **Post-OCR Error Correction:**

   Integrating rule-based correction, regex validation, or a lightweight language model can compensate for character mis-detections and produce cleaner license plate outputs.

10. **Integration with Intelligent Transportation Systems (ITS):**

    Future implementations can include backend pipelines for parking automation, violation detection, vehicle entry–exit logs, and real-time monitoring dashboards.

References

[1] B. Buleu, R. Robu, and I. Filip, “License Plate Recognition Dataset — YOLO Format, Version 11,” *Roboflow Universe*, 2025. [Online]. Available: https://universe.roboflow.com/roboflow-universe-projects/license-plate-recognition-rxg4e/dataset/11

[2] Ultralytics, “YOLO Performance Metrics — Class-wise Precision, Recall, mAP Explained,” *Ultralytics Documentation*, 2025. [Online]. Available: https://docs.ultralytics.com/guides/yolo-performance-metrics/

[3] PaddlePaddle Contributors, “PaddleOCR: Multilingual Optical Character Recognition Toolkit,” *PaddleOCR Documentation*, 2025. [Online]. Available: https://paddlepaddle.github.io/PaddleOCR

[4] Buleu, B.; Robu, R.; Filip, I. A Deep Learning-Based System for Automatic License Plate Recognition Using YOLOv12 and PaddleOCR. *Appl. Sci.* **2025**, *15*(14), 7833. https://doi.org/10.3390/app15147833

[5] *YOLOv8: Next-Generation Vision AI Models*.
Available online: https://docs.ultralytics.com/models/yolov8