License Plate Detection and Recognition: a Two-Stage Approach using YOLOv9 and EasyOCR

1st Philippe Lizotte [ID: 40261140]
Gina Cody School of ECS, CSSE Dept.
Concordia University, Montréal, Québec, Canada
p_lizot@live.concordia.ca

2nd Marmik Patel [ID: 40201462]
Gina Cody School of ECS, CSSE Dept.
Concordia University, Montréal, Québec, Canada
p_marmik@live.concordia.ca

3rd Phuc Thien Tran [ID: 40017354]
Gina Cody School of ECS, CSSE Dept.
Concordia University, Montréal, Québec, Canada
phucthientran@gmail.com

4th Giuliano Verdone [ID: 40252190]
Gina Cody School of ECS, CSSE Dept.
Concordia University, Montréal, Québec, Canada
g_verdon@live.concordia.ca

Abstract

We seek to tackle the ongoing challenge of detecting license plate using Deep Learning by implementing a two-step pipeline. First, a YOLOv9 model is used to find and isolate license plates from images provided from a custom datasets comprised of license plates from several regions. Second, an EasyOCR model is used to read these license plates by employing an optimized character recognition algorithm. This paper discusses the detailed process of developing and implementing these models, followed by a discussion of the results and inferred findings from training and testing this two-phase algorithm on the aforementioned dataset.

Index Terms

YOLOv9, License Plate Detection, OCR, Character Recognition, Computer Vision, Deep Learning

I. Introduction

License plate detection and recognition remain challenging due to their accuracy being impacted by diverse lighting conditions, obstructions, and regional plate variations. Addressing these issues has direct benefits towards plate detection's applications in traffic management, parking detection, and even for charging tolls when crossing through points of interest. Traditional computer vision methods often fall short under these noise-related conditions, requiring more robust deep learning approaches. Advancements in object detection, namely the popular YOLO (You Only Look Once) family, have improved both speed and accuracy. Despite these improvements, a model's performance for license plate and character recognition still depends on the quality of the training dataset. As such, this study focuses on training a YOLOv9 model on two datasets: a public Kaggle and RoboFlow dataset comprised of roughly 10,000 international license plates, as well as a dataset of 1,600 images covering license plates from Quebec, Ontario, California, and New York to optimize detection accuracy in a North American context. Due to the model's limitation on character recognition, we made use of EasyOCR to infer the characters.

II. Related Work

The authors of YOLOv9, in particular, introduced architectural enhancements to the family that boost detection performance, environmental robustness, and computational efficiency [1]. Dataset quality is crucial for license plate recognition, as diverse, high-quality data improves generalization and accuracy [2], [3]. Key factors include annotation accuracy—critical for detecting small plates [4]—and data diversity across angles, lighting, obstructions, and plate styles, especially when sourced from multiple regions [5]. This project uses real-world images from varied North American settings. While synthetic data (e.g., GAN-generated) can supplement limited datasets [6], it was unnecessary here due to sufficient real-world coverage.

A. Comparative Analysis Overview

YOLOv9 introduces significant enhancements over YOLOv8, notably through architectural advancements such as Programmable Gradient Information (PGI) and the Generalized Efficient Layer Aggregation Network (GELAN). These improvements enable YOLOv9 to achieve higher accuracy and more efficient feature learning compared to YOLOv8 [1]. While YOLOv8 models exhibit strong performance and slightly faster inference speeds, YOLOv9 variants provide better generalization across diverse conditions—particularly beneficial for complex detection tasks like license plate recognition. In our work, YOLOv9c offers an optimal balance of accuracy and computational efficiency, making it ideal for real-time detection scenarios with limited resources.

B. Performance Metrics Table

The table below summarizes key performance metrics for different variants of YOLOv8 and YOLOv9, as evaluated on the COCO dataset:

TABLE I
PERFORMANCE METRICS FOR YOLOV8 AND YOLOV9 VARIANTS ON THE COCO DATASET

Model	mAP@0.5:0.95 (%)	Parameters (M)	FLOPs (B)	Inference Speed (ms)
YOLOv8n	37.3	3.2	8.7	1.47
YOLOv8s	44.9	11.2	28.6	2.66
YOLOv8m	50.2	25.9	78.9	5.86
YOLOv8l	52.9	43.7	165.2	9.06
YOLOv8x	53.9	68.2	257.8	14.37
YOLOv9t	38.3	2.0	7.7	2.30
YOLOv9s	46.8	7.1	26.4	3.54
YOLOv9m	51.4	20.0	76.3	6.43
YOLOv9c	53.0	25.3	102.1	7.16
YOLOv9e	55.6	57.3	189.0	16.77

Source: Ultralytics Documentation [1].

C. Architectural Enhancements in YOLOv9

YOLOv9 brings several key architectural improvements:

- **Programmable Gradient Information (PGI):** Enhances gradient flow, leading to more efficient learning.
- **Generalized Efficient Layer Aggregation Network (GELAN):** Optimizes feature aggregation for improved detection accuracy.
- **Depthwise Convolutions and C3Ghost Modules:** Reduce computational complexity while maintaining performance.

For additional details, refer to https://arxiv.org/html/2409.07813v1 [11].

D. Performance in License Plate Recognition

In cloud-based license plate recognition (LPR) tasks, YOLOv8 achieved a testing accuracy of 78%, while YOLOv9 reached 70%. Despite the higher accuracy of YOLOv8 in this scenario, YOLOv9 demonstrated consistent performance across varying conditions—an indicator of better generalization [2]. Based on these observations and efficiency metrics, the YOLOv9c model was chosen for license plate detection.

E. Comparative Analysis of YOLOv9 Model Variants

The YOLOv9 series comprises several variants designed for different deployment scenarios:

- **YOLOv9t (Tiny):** Designed for highly constrained environments, with about 2 million parameters and an mAP@0.5 of 38.3%.
- **YOLOv9s (Small):** Approximately 7.2 million parameters, achieving an mAP@0.5 of 46.8%.
- **YOLOv9m (Medium):** Balances performance and efficiency with around 20.1 million parameters and an mAP@0.5 of 51.4%.
- **YOLOv9c (Compact):** Offers an optimal balance with roughly 25.5 million parameters and an mAP@0.5 of 53.0%.

- **YOLOv9e (Enhanced):** The largest variant with about 58.1 million parameters and a 55.6% mAP@0.5, though it requires more computational resources.

Source: https://arxiv.org/html/2409.07813v1 [11].

F. Rationale for Selecting YOLOv9c in License Plate Detection

Several factors influenced the selection of YOLOv9c:

1) **Efficient Performance:** YOLOv9c delivers competitive detection accuracy with approximately 42% fewer parameters and 21% less computational overhead compared to heavier models like YOLOv7-AF [1], [8].
2) **Balanced Accuracy and Inference Speed:** It strikes an optimal trade-off between performance and speed, essential for real-time applications [2].
3) **Innovative Architecture:** The incorporation of PGI and GELAN supports robust generalization across various deployment scenarios [3].

G. Application in License Plate Detection

YOLOv9c has been effectively applied in license plate detection systems, often integrated with OCR tools like EasyOCR for character recognition. This combination has demonstrated high precision and recall in real-time scenarios [4].

III. Proposed Methodology

The methodology follows a two-stage pipeline: (1) training YOLOv9-c on a curated dataset for license plate detection under varied conditions, and (2) applying EasyOCR to extract alphanumeric characters from detected plates. The project progressed through five phases: setup, literature review, data preparation, model implementation, and final analysis. While the Agile workflow was initially considered, the four-month timeline favored a structured Waterfall model, better suited to projects with defined scopes and minimal change. Given the maturity of detection models, iterative planning was deprioritized in favor of post-hoc enhancements via self-learning and follow-up investigations.

A. Data Collection and Processing

In this study, the primary focus is on detecting license plates using the YOLOv9-c model, followed by Optical Character Recognition (OCR) for extracting the alphanumeric content of the license plates. A custom dataset was used which consisted of 1,600 images of license plates from Quebec, Ontario, California, and New York. The dataset was captured and curated by Zahra Ebrahimi. It was made available by the Centre for Pattern Recognition and Machine Intelligence (CENPARMI) alongside Dr. C. Y. Suen, the Director of CENPARMI [7]. It is used under terms of the CENPARMI Database Agreement.

1) Image Preprocessing: In the course of assembling a robust dataset for license plate detection, the project team examined multiple sources. Four potential sources were identified: Concordia's internal dataset (provided by the TA), LicensePlateMania, Roboflow's public datasets, and Kaggle. The following subsections detail the suitability assessments, decisions, and acquisition methods for each source.

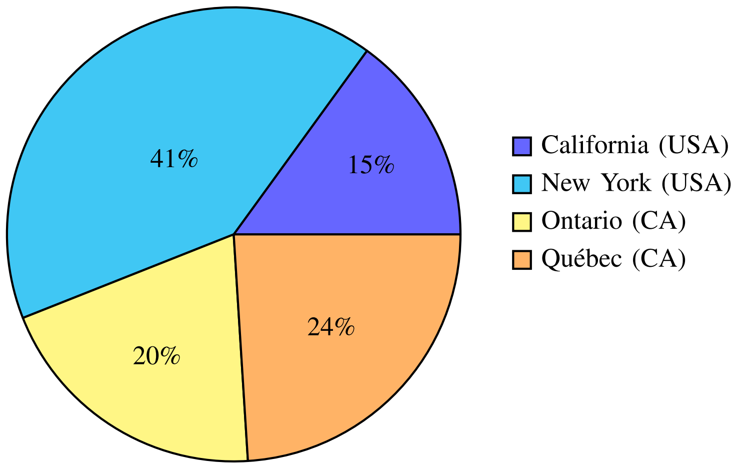
TABLE II
License Plate Dataset Sources

Source	**Description**	**Use/Access**
Concordia Internal and CENPARMI [7]	1,600 labeled images from NY, CA, ON, and QC, organized by region.	Accessed via NDA.
LicensePlateMania	50,000+ global images with general labels.	Not used due to €70 license fee.
Roboflow (Public) [8]	Open-source datasets for license plate detection.	Used under public license.
Kaggle	2,500 unlabeled images.	Scraped and labeled using a custom tool.

TABLE III
CENPARMI DATA DISTRIBUTION ACROSS REGIONS, IN MB.

Region	Test	Train	Total
California, USA	1.13	5.18	6.32
New York, USA	9.38	7.85	17.2
Ontario, CA	4.25	3.97	8.22
Québec, CA	6.66	3.57	10.2

Fig. 1. CENPARMI Data Distribution by North American Region (in MB)



2) Data Processing: To prepare the collected images for license plate detection using YOLOv9, the project team performed several pre-processing steps to enhance the quality and structure of the dataset.

Image Duplication and Grayscale Conversion: As a preliminary experiment to assess the potential impact of color on detection accuracy, each image in the dataset was duplicated and converted to grayscale. This enabled a comparative analysis to determine whether color information significantly affects model performance. Both the original (color) and grayscale versions were retained, allowing further investigation if color features were later found to be beneficial or necessary.

Dataset Splitting: The dataset was partitioned into three distinct subsets to facilitate training, validation, and testing:

- **Training Set (50%):** Used to update the parameters of the YOLOv9 model during training. This subset includes a wide variety of license plate styles and environmental conditions.
- **Validation Set (25%):** Utilized during training to monitor performance on previously unseen data. It supports hyperparameter tuning and early stopping to mitigate overfitting.
- **Testing Set (25%):** Employed post-training to provide an unbiased evaluation of the model's generalization ability on new, unseen data.

3) Conforming to YOLOv9 Structure: To align with the input requirements of YOLOv9, several Python scripts were developed to automate the dataset preparation process. These scripts handled the formatting of image files, bounding box annotations, and corresponding label files into the folder hierarchy expected by the YOLOv9 framework.

B. Model Setup

First, the YOLOv9 model is first trained to detect license plates within various environmental conditions, considering variations in plate design, lighting, obfuscations, and angles. Post detection, an OCR engine gets applied to the localized plates to extract the characters. This two-step approach combines the accuracy of YOLOv9 for object detection with the precision of OCR for character recognition.

1) Implementation: Our implementation consists of two main components: license plate detection using YOLOv9 and character recognition via EasyOCR.

License Plate Detection (YOLOv9): YOLOv9 was trained on a custom dataset of 1,600 annotated license plate images from Quebec, Ontario, California, and New York, sourced from the Centre for Pattern Recognition and Machine Intelligence (CENPARMI). To ensure diversity, images were duplicated and partially converted to grayscale, enabling analysis of color's impact on detection.

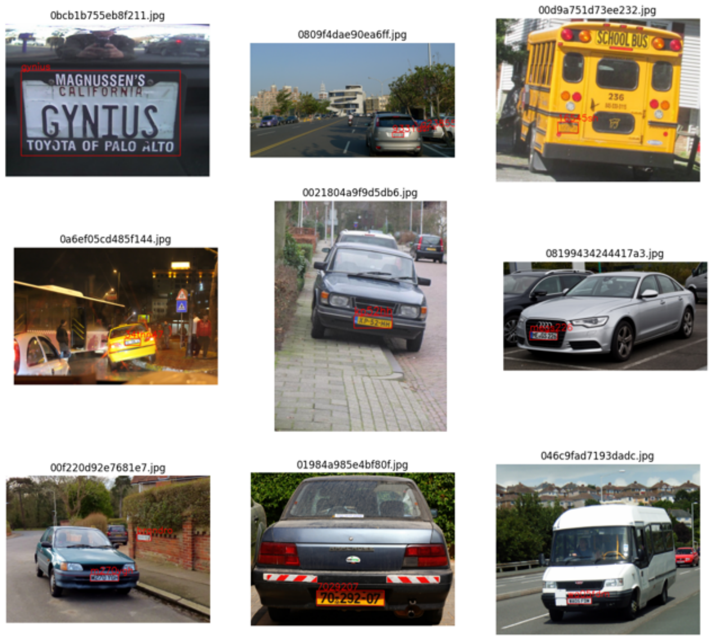


Fig. 2. Validation results from YOLOv9-based plate detection.

Character Recognition (EasyOCR): Once plates were detected, the corresponding regions were cropped and processed using EasyOCR, which supports variable fonts, orientations, and lighting. Preprocessing steps included contrast enhancement and noise reduction to improve OCR reliability.

The YOLOv9–EasyOCR pipeline demonstrated a robust, real-world-ready solution for license plate recognition, balancing high detection accuracy with dependable character extraction.

IV. Results and Discussion

For the training process of our model, we leveraged the freely available computational resources provided by Google Colab. Specifically, our models were trained using the NVIDIA Tesla T4 GPUs for a balance between performance and memory efficiency. In addition to cloud-based training, local development and testing were also performed on a machine equipped with an NVIDIA RTX 3070 GPU using CUDA. With this, work was accomplished faster during model development and debugging, specifically for adjustments to model parameters. The goal was to select hardware that's able to perform well when given deep learning tasks.

A. Evaluation Metrics

To evaluate the performance of our models, we measured and recorded the following three metrics:

Mean Average Precision (mAP): a single-number summary of the precision-recall curve across all classes and different levels of intersection-over-union (IoU) thresholds. A higher mAP indicates that the model is accurately detecting objects and drawing bounding boxes that closely align with the truth.

Precision: the proportion of true positive predictions out of all positive predictions made by the model. High precision indicates that when the model identifies a license plate, it is usually correct, meaning there are few false positives.

Recall: the proportion of actual positives that were correctly identified by the model. A high recall score suggests that the model successfully detects most of the license plates in the dataset, minimizing false negatives.

Based on these three metrics, comparative analysis against publicly available models would allow us to validate that our model generalizes well to unseen data and does not overfit or exhibit bias toward our training dataset.

B. Detection and Recognition

Some character recognition issues emerged across license plates from all regions. For example, Figure 3 shows how watermarks interfered with detection, while Figure 4 highlights confusion between visually similar characters such as ‘8’ and ‘B’.

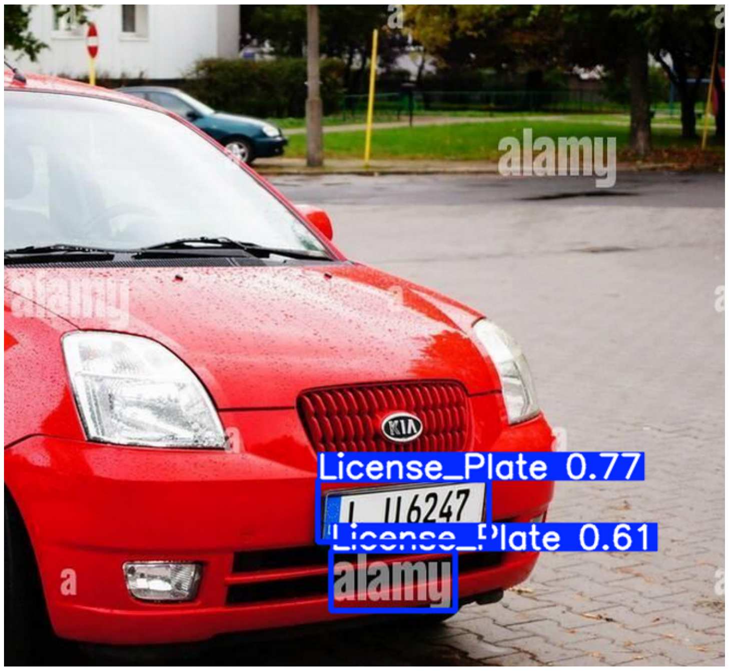


Fig. 3. Model confusion due to watermark interference.

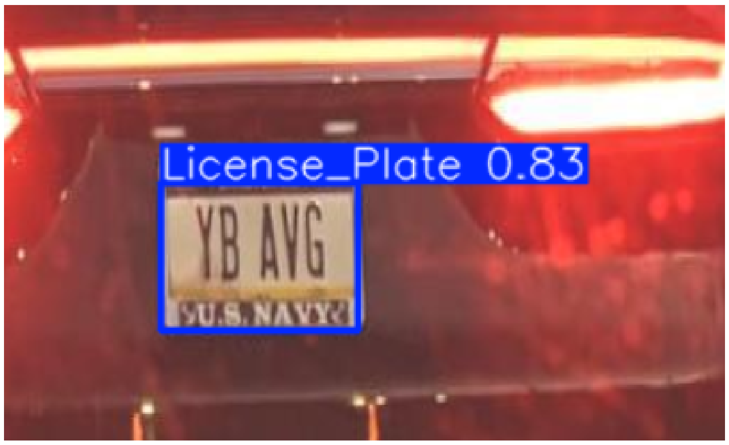
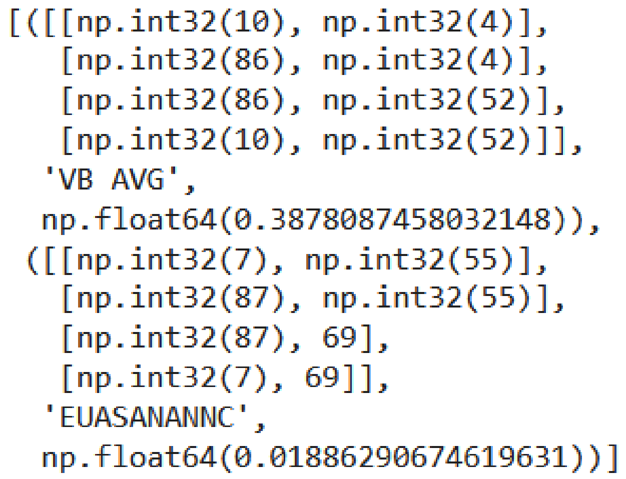


Fig. 4. Misclassification between ‘8’ and ‘B’ on a U.S. Navy plate.

In Figure 4, the model misread a ‘B’ as ‘8’ due to poor lighting, glare, and low resolution—common challenges in real-world frames. These visual ambiguities often degrade character boundaries and increase recognition errors.

To better understand these limitations, we visualized the character recognition outputs. As shown in Figure 5, the cropped plate is passed to the OCR module, which predicts bounding boxes and decodes the character sequence.



```
[([[np.int32(10), np.int32(4)],
   [np.int32(86), np.int32(4)],
   [np.int32(86), np.int32(52)],
   [np.int32(10), np.int32(52)]],
  'VB AVG',
  np.float64(0.3878087458032148)),
 ([[np.int32(7), np.int32(55)],
   [np.int32(87), np.int32(55)],
   [np.int32(87), 69],
   [np.int32(7), 69]],
  'EUASANANNC',
  np.float64(0.01886290674619631))]
```

Fig. 5. Character predictions with bounding boxes.

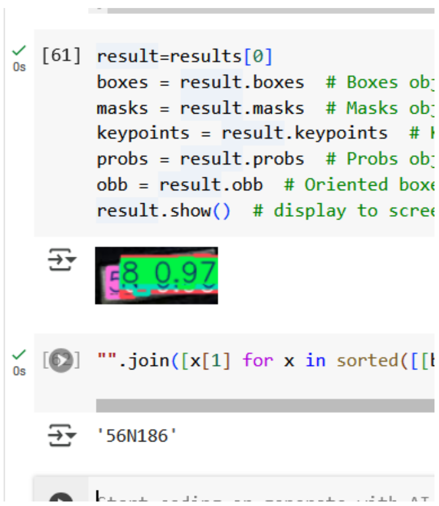
Although segmentation performed reasonably well, the decoded string `VB AVG` was incorrect; the true text was `YB AVG`. Similarly, stylized text (e.g., ”U.S. NAVY”) was misread as `EUASANANNC` with low confidence (≈ 0.018). This illustrates the model’s difficulty with small fonts, non-standard formats, and government/military plates.

Such errors, while subtle, can significantly impact downstream tasks requiring precise identification. Improvements could include integrating context-aware post-processing, common-format constraints, and training with more edge-case samples (e.g., ‘B’ vs. ‘8’, ‘O’ vs. ‘0’, ‘I’ vs. ‘1’). Another significant challenge is intra-class variation in license plate designs across different regions. While our

OCR model was trained on 2,000 plate crops primarily sourced from Quebec, Ontario, California, and New York, this limited scope restricts the model's ability to generalize to international formats. Global license plates varied not only in language and character sets but also in glyph/font styles, alignment (e.g., one-line vs. two-line), and embedded symbols.

C. Visualization

To evaluate end-to-end performance, we visualized outputs from both detection and recognition stages.



```
[61] result=results[0]
     boxes = result.boxes  # Boxes ob
     masks = result.masks  # Masks ob
     keypoints = result.keypoints  # K
     probs = result.probs  # Probs ob
     obb = result.obb  # Oriented boxe
     result.show()  # display to scre
```

```
"".join([x[1] for x in sorted([[
```

```
'56N186'
```

Fig. 6. Character-level recognition output.

In Figure 6, each character is detected and labeled with confidence scores. Here, the model correctly reconstructs `56N186`, with '8' identified at 0.97 confidence, indicating reliable localization and decoding.

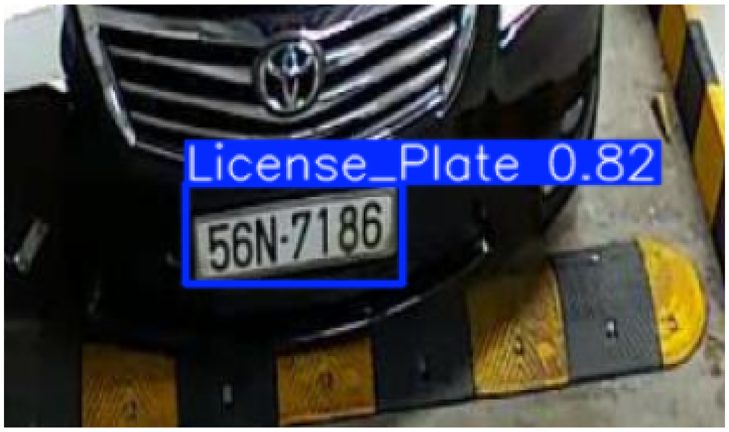


Fig. 7. License plate detection results using YOLOv9.

Figure 7 shows the initial detection stage, where YOLOv9 identifies the license plate with a bounding box and 0.82 confidence, despite lighting issues, angle shifts, and partial occlusion.

D. Benchmarking Model Performance

To evaluate the robustness and versatility of our license plate recognition system, we compared our detection and OCR models against several publicly available alternatives. In particular, we benchmarked our detection model against four alternate detectors: `amykun-qoz6t`, `ev-dshfb`, `ayush-thakre`, and `allen-liu`. Our OCR component was compared with `joey-seyrt` and `License Plates Workspace Assist`.

This approach involved running our model on each competitor's dataset (when publicly available) and measuring performance using standard metrics such as average precision, recall, and mAP score. We then compiled these results and compared them to the performance reported by the respective model owners on their own datasets.

Despite achieving strong performance metrics (particularly on our in-house dataset), our model exhibited a significant drop in accuracy on certain external datasets, especially those used by amykun-qoz6t, ev-dshfb, ayush-thakre-ohczr, joey-seyrt, and License Plates Workspace Assist. A closer examination of these discrepancies revealed several contributing factors:

- **Weather Variability**: Adverse conditions such as rain, snow, fog, or poor lighting led to lower-quality images containing additional visual noise.
- **Photo Quality**: Issues such as broken pixels, low resolution, or overly compressed images adversely affected detection and OCR performance, especially if the training data did not account for these scenarios in sufficient quantity.

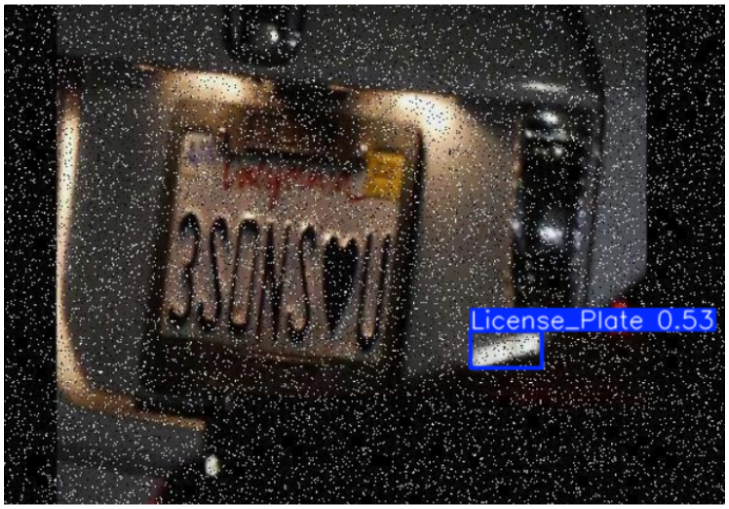


Fig. 8. Unsuccessful detection (confidence score: 0.53) of a rear vehicle plate in low-light and high-noise conditions.

- **Image Rotations**: Our model maintained an accuracy of around 95% when the license plate plane was nearly parallel to the camera's viewpoint. However, angles exceeding 45 degrees sharply reduced detection confidence, resulting in missed or partially recognized plates.

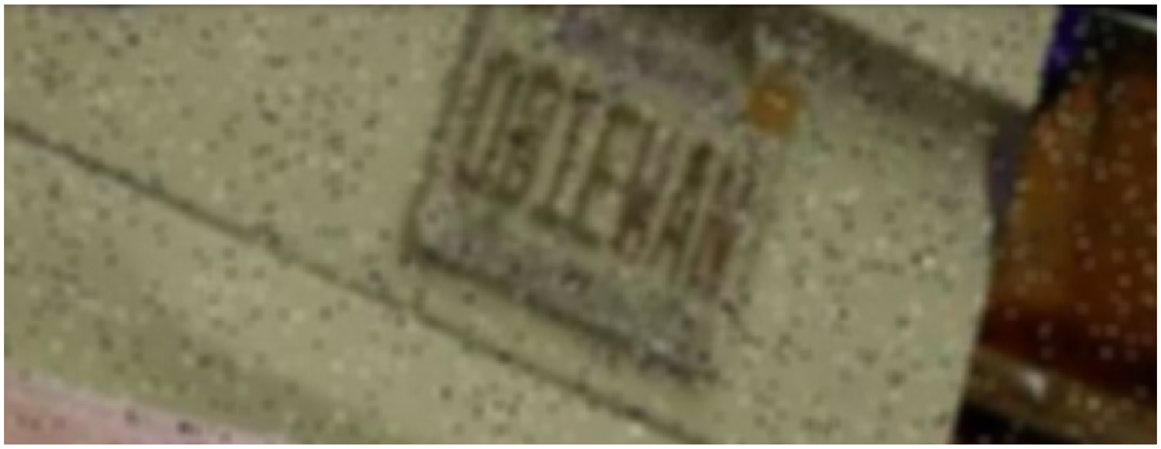


Fig. 9. Challenging license plate detection scenario involving rotation and intentional noise.

This was a shortcoming that demonstrates the model's need for more robust perspective invariance or data augmentation strategies that include rotation.

- **Conformity of Plate Designs**: Our system was primarily trained on two-line license plates. Consequently, it struggled when encountering plates with unusual formatting or character arrangements. Because the external datasets included plate styles or formats that diverged from our training distribution, the system failed to detect or accurately read these non-standard configurations.

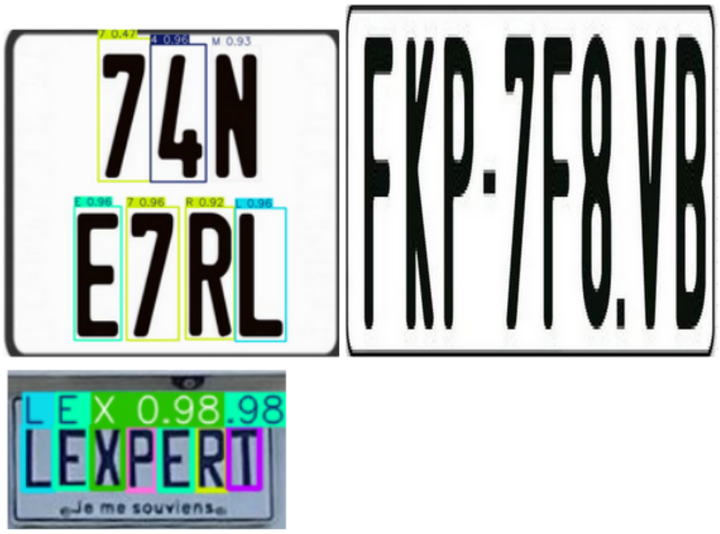


Fig. 10. Comparison between two-line and one-line license plate readings.

- **Dataset Formatting**: External datasets often feature label formats, class definitions, or annotation structures that differ from our own. Such inconsistencies can confuse the model, leading to mislabeling or missed detections, which, in turn, negatively impacts overall accuracy.

The tables below summarize the performance metrics for our model compared to selected publicly available alternatives.

TABLE IV
COMPETITOR PLATE DETECTION MODELS

Model Name	Model URL	Precision	Recall	mAP50
amykun-qoz6t	https://universe.roboflow.com/amykun-qoz6t/license-plate-recognition-8fvub/model/1	0.915	0.04	0.48
ev-dshfb	https://universe.roboflow.com/ev-dshfb/license-plate-w8chc/dataset/1	0.25	0.16	0.16
ayush-thakre	https://universe.roboflow.com/ayush-thakre-ohczr/xyz-oftze/dataset/1/download/yolov9	0.74	0.44	0.57
allen-liu	https://universe.roboflow.com/allen-liu-bxf3t/car_license_position/dataset/1	0.98	0.95	0.98

TABLE V
COMPETITOR OCR MODELS

Model Name	Model URL	Precision	Recall	mAP50
joey-seyrt (OCR)	https://universe.roboflow.com/joey-seyrt/test-5hvrv	0.0	0.0	0.0
License Plates Workspace Assist	https://universe.roboflow.com/license-plates-workspace-assist/license-plates-valid	0.51	0.34	0.29

TABLE VI
OUR MODEL PERFORMANCE METRICS

Model Component	Precision	Recall	mAP50	mAP50-95	Fitness
Plate Detection Model	0.987	0.959	0.982	0.748	0.771
OCR Model	0.948	0.944	0.966	0.815	0.830

While `allen-liu` achieves near-perfect metrics (precision 0.98, recall 0.95, mAP50 0.98), `amykun-qoz6t` shows high precision (0.915) but extremely low recall (0.04), meaning it rarely detects plates despite accurate predictions when it does. In contrast, `ev-dshfb` and `License Plates Workspace Assist` underperform across all metrics, and `ayush-thakre` offers moderate results. Notably, the OCR model `joey-seyrt` registers no detections (it can be said that it's essentially ineffective). Despite our model not being able to outperform every single commercial model listed above, our work did reveal trade-offs between precision and recall in license plate detection.

E. Summary of Limitations

Despite promising results, our system faces several limitations. Environmental challenges such as adverse weather, low-light conditions, and significant image noise can reduce detection accuracy. The model also struggles with rotated or non-standard license plate formats, leading to occasional misclassifications. Additionally, the limited, region-specific dataset may affect generalizability to a broader context. Moreover, these issues can be attributed in part due to the steep learning curve our team encountered while mastering YOLO, computer vision fundamentals, and the intricacies of training and running deep learning models.

V. CONCLUSION AND FUTURE WORK

To conclude, this study concentrates on the application of a two-step pipeline involving YOLOv9 and EasyOCR-based to create a model for license plate detection and character recognition across diverse environmental conditions. YOLOv9 did exhibit accuracy in detecting license plates in controlled conditions. However its performance was negatively impacted in scenarios involving significant image noise, namely watermarks or partial obstructions. EasyOCR provided a reliable solution for character recognition post-detection but similarly showed sensitivity to dataset quality. Building on our current findings and the lessons learned from testing across diverse datasets, our future research agenda focuses on two main objectives: improving model accuracy and expanding the scope of model deployment. We plan to diversify the OCR training dataset by including annotated images that continue to be comprised of multiple countries and plate conventions. This would improve the robustness of character recognition and reduce misclassification errors due to unfamiliar formats. Improving the model's resilience to noise would undoubtedly enhance its real-world applicability. Thus, this research contributes to aligning recent advances in deep learning with the practical implementation of license plate detection and recognition systems.

VI. Acknowledgement

The authors wish to acknowledge Dr. Yaser Esmaeili Salehani and Zahra Ebrahimi from Concordia University's Department of Computer Science and Software Engineering (CSSE) for their guidance throughout this research project. It was conducted as a part of coursework from COMP 425/6341 - Computer Vision, for the Winter 2025 semester.

A. References

References

[1] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, "YOLOv9: Learning What You Want to Learn Using Programmable Gradient Information," Cornell University, Feb. 29, 2024. [Online]. Available: https://arxiv.org/abs/2402.13616. [Accessed: Feb. 27, 2025].

[2] Z. Peng, Y. Gao, S. Mu, and S. Xu, "Toward Reliable License Plate Detection in Varied Contexts: Overcoming the Issue of Undersized Plate Annotations," *IEEE Trans. Intell. Transp. Syst.*, Jul. 15, 2024. [Online]. Available: https://ieeexplore.ieee.org/document/10598829. [Accessed: Feb. 27, 2025].

[3] J. L. C. Weng, H. A. Karim, and N. Aldahoul, "An Evaluation of Various Pre-trained Object Detection Models for Complex License Plate," ResearchGate, Jul. 2024. [Online]. Available: https://www.researchgate.net/publication/387923315_An_Evaluation_of_Various_Pre-trained_Object_Detection_Models_for_Complex_License_Plate. [Accessed: Feb. 27, 2025].

[4] S. Agrawal, "Global License Plate Dataset," Cornell University, Mar. 22, 2024. [Online]. Available: https://arxiv.org/abs/2405.10949. [Accessed: Feb. 27, 2025].

[5] T. Sasaki, K. Morita, and T. Wakabayashi, "License plate recognition using 3D rotated character recognition and deep learning," ResearchGate, Nov. 2022. [Online]. Available: https://www.researchgate.net/publication/366887473_License_plate_recognition_using_3D_rotated_character_recognition_and_deep_learning. [Accessed: Feb. 27, 2025].

[6] Y.-H. Cheng and P.-Y. Chen, "Research on High-Fidelity License Plates Generated Using Generative Adversarial Network (GAN) Technology for Vehicle Recognition Systems," ACM Digital Library, May 30, 2024. [Online]. Available: https://dl.acm.org/doi/10.1145/3651781.3651806. [Accessed: Feb. 27, 2025].

[7] Centre for Pattern Recognition and Machine Intelligence, "About - CENPARMI," Concordia University. [Online]. Available: https://www.concordia.ca/research/cenparmi/about.html. [Accessed: Feb. 27, 2025].

[8] Augmented Startups, "Vehicle Registration Plates Dataset," Roboflow. [Online]. Available: https://universe.roboflow.com/augmented-startups/vehicle-registration-plates-trudk. [Accessed: Feb. 27, 2025].

[9] Ultralytics, "Model Comparison: YOLOv8 vs YOLOv9 for Object Detection," [Online]. Available: https://docs.ultralytics.com/compare/yolov8-vs-yolov9/.

[10] Asaju, C. B., Owolawi, P. O., Tu, C., & Van Wyk, E., "Cloud-Based License Plate Recognition: A Comparative Approach Using YOLO Versions 5,7,8 and 9 Object Detection," MDPI Information, 16(1), 57, 2024, [Online]. Available: https://www.mdpi.com/2078-2489/16/1/57.

[11] Yaseen, M., "What is YOLOv9: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector," [Online]. Available: https://arxiv.org/html/2409.07813v1.

[12] LearnOpenCV, "YOLOv9: Advancing the YOLO Legacy," [Online]. Available: https://learnopencv.com/yolov9-advancing-the-yolo-legacy/.