DOI: 10.2478/ias-2026-0016 • JIAS • 21(6) • 2026 • 318–333

Article

From YOLO Models to Embedded Deployment: A Blockchain-Enabled ANPR System on Jetson and Raspberry Pi

Nesrine Affes [1], Jalel Ktari [1,*], and Habib Hamam [2,3,4,5]

1 CES-lab, National Engineering School of Sfax, University of Sfax, Sfax, 3038 Sfax, Tunisia; nesrine.affes@enis.tn
2 Faculty of Engineering, University of Moncton, NBE1A3E9, 18, avenue Antonine-Maillet, Canada; habib.hamam@umoncton.ca
3 School of Electrical Engineering, University of Johannesburg, Johannesburg,2121, South Africa
4 International Institute of Technology. and Management, Mindoubé 2, Libreville, BP. 9985, Gabon
5 Training Center: Bridges for Academic Excellence - Spectrum, Tunis, 1002, Tunisia
* Correspondence author: jalel.ktari@enis.tn

Abstract: This paper presents a secure blockchain-enabled Automatic Number Plate Recognition (ANPR) framework for Intelligent Transportation Systems (ITS), combining lightweight YOLO-based license plate detection, OCR-based character recognition, and Ethereum smart contracts for tamper-resistant vehicle data management. The study investigates the inference performance of recent object detection architectures, namely YOLOv11, YOLOv12, and YOLO26, on Raspberry Pi 5 and NVIDIA Jetson Nano edge devices. Furthermore, optimizations for ONNX and TensorRT are explored to enhance inference efficiency on embedded hardware. Experimental results demonstrate complementary strengths among the evaluated models. YOLOv11n achieved the best overall detection performance, obtaining the highest recall (0.945), F1-score (0.9645), and mAP@0.5:0.95 (0.692), while YOLOv11n and YOLOv12n achieved identical precision (0.985) and mAP@0.5 (0.963). YOLO26 demonstrates the best deployment efficiency, reducing inference latency to 33.4 ms on Raspberry Pi (ONNX) and 38.9 ms on Jetson Nano, making it suitable for edge applications. Model optimization significantly accelerates inference, reducing YOLOv11 latency on Raspberry Pi from 98.2 ms to 38.3 ms after ONNX conversion. The results demonstrate that the proposed framework effectively balances detection accuracy, computational efficiency, and data security, making it a promising solution for smart parking and next-generation ITS applications.

Keywords: ANPR; YOLO; easyOCR; NVIDIA Jetson; Raspberry Pi; Ethereum.

1. Introduction

ANPR has become a key technology for allowing smart mobility, traffic automation, and secure vehicle monitoring due to the quick development of ITS and the Internet of Vehicles (IoV). ANPR systems are frequently employed in contemporary smart city infrastructures for tasks including access control, parking management, toll collecting, and law enforcement. Low-latency performance, excellent precision, and resilience in the face of difficult environmental factors like shifting lighting, occlusions, motion blur, and intricate metropolitan backgrounds are necessary for these applications. It is proven that traditional ANPR algorithms based on conventional image processing techniques have limitations in terms of scalability and stability. Thus, deep learning-based solutions, especially object detection frameworks such as YOLO, have been adopted for real-time license plate detection.

Deploying ANPR systems in real-world settings is still difficult despite tremendous advancements because of limitations with low-latency, and system scalability. Recent developments in lightweight deep neural networks have made it possible to deploy them effectively on edge computing platforms like

NVIDIA Jetson devices and Raspberry Pi. In parallel, blockchain technology, particularly Ethereum-based smart contracts, has emerged as a powerful solution to enhance data integrity, transparency, and security in ITS applications. Blockchain eliminates single points of failure and guarantees reliable data transmission between vehicles and infrastructure by storing vehicle-related events in a decentralized, impenetrable ledger.

In this work, we propose a secure and efficient edge-based ANPR framework that integrates YOLO-based object detection for license plate localization, EasyOCR/PyTesseract for character recognition, and Ethereum blockchain for secure event logging and transaction management. The system was designed and evaluated on embedded platforms, including the Raspberry Pi 5 and NVIDIA Jetson Nano, to assess inference performance, computational efficiency, and deployment feasibility under resource-constrained hardware conditions. The objective is to bridge the gap between high-accuracy deep learning models and practical ITS deployment by integrating efficient edge inference with blockchain-based secure and tamper-resistant data management.

The main contributions of this paper are summarized as follows:

(i) A lightweight and robust ANPR framework based on recent YOLO architectures (YOLOv11n, YOLOv12n, and YOLO26n) combined with OCR for accurate low-latency license plate detection and recognition under diverse operating conditions.
(ii) A comprehensive comparative evaluation of multiple YOLO generations across heterogeneous edge-computing platforms, namely Raspberry Pi 5 and NVIDIA Jetson Nano 2GB, considering both detection accuracy (Precision, Recall, mAP) and deployment efficiency (latency, memory usage, CPU/GPU utilization).
(iii) An investigation of model optimization strategies for edge AI deployment through ONNX and TensorRT acceleration, highlighting the trade-offs between detection performance and computational efficiency on resource-constrained hardware.
(iv) A secure and decentralized architecture integrating Ethereum blockchain technology to provide tamper-resistant storage of vehicle events, ensuring data integrity, traceability, and trust for intelligent transportation and smart parking applications.

The remainder of this paper is organized as follows: Section2 describes the state of the art. Section3 proposed methodology including the dataset and evaluation metrics. Section 4 presents the experimental results, covering detection performance, OCR evaluation metrics, Performance evaluation on edge platforms and blockchain transaction cost assessment. Sections 5 and 6 discuss the findings, conclude the paper, and outline directions for future work.

2. Literature Review

This section reviews the main advances in ANPR, deep learning-based detection and recognition methods, edge computing deployment, and blockchain-enabled intelligent transportation systems. Early ANPR systems relied on traditional image processing and OCR techniques, which were highly sensitive to illumination variations, motion blur, and complex backgrounds, resulting in limited robustness in real-world environments. The advent of deep learning, particularly Convolutional Neural Networks (CNNs), significantly improved license plate detection and recognition performance by enabling automatic feature extraction and enhanced generalization capabilities. However, many early deep learning solutions required substantial computational resources, restricting their deployment to cloud or high-performance computing environments.

Recent research has increasingly adopted end-to-end ANPR frameworks based on YOLO object detection architectures, which provide an effective balance between detection accuracy and processing speed [1]. Combined with modern OCR engines, these approaches achieve superior performance compared with conventional methods while supporting real-time operation. Despite these advances, challenges remain regarding efficient deployment on resource-constrained edge devices and the secure management of vehicle data, motivating the development of integrated ANPR solutions that combine lightweight deep learning models, edge computing, and blockchain technologies.

In parallel, research efforts have increasingly shifted toward edge computing and embedded deployment scenarios due to the growing demand for low-latency, privacy-preserving, and energy-efficient solutions. Platforms such as Raspberry Pi and NVIDIA Jetson have become widely adopted for real-time deep learning inference on constrained hardware. In particular, Jetson devices benefit from integrated GPU acceleration and support for CUDA and TensorRT, making them highly suitable for deploying optimized deep learning models.

To further enhance inference efficiency on embedded systems, various optimization strategies have been explored, including model quantization, conversion to ONNX format, and TensorRT-based acceleration. These techniques have been shown to significantly improve the execution speed of YOLO-based models while preserving acceptable levels of accuracy, thereby enabling their use in smart mobility and real-time ITS applications.

Author in [2] presents a real-time ANPR system fully deployed on the NVIDIA Jetson Orin Nano using edge AI principles. It integrates a YOLOv8-based model for license plate detection with an OCR engine (EasyOCR) for character recognition. To achieve low-latency inference, the model is optimized using ONNX and TensorRT with FP16 precision. The entire pipeline operates on-device without cloud dependency, ensuring privacy and fast processing. Experimental results show reliable real-time performance with good accuracy, demonstrating the feasibility of edge-based ANPR systems for intelligent transportation applications. However, the approach has some limitations, including dependence on dataset quality and the use of general-purpose OCR which may reduce accuracy for complex plate formats.

The work presented in [3] emphasizes the importance of license plate recognition in supporting traffic law enforcement and reducing road accidents, particularly in multilingual countries. It proposes a deep learning-based approach for recognizing Arabic and Latin license plates, with a focus on regions such as Iraq and Malaysia. The study highlights the role of Automatic Plate Detection and Recognition Technology in various security applications, including police surveillance, traffic monitoring, and parking management. To improve recognition accuracy, transfer learning techniques are employed using models such as DenseNet121, MobileNetV2, and NASNetMobile, where DenseNet121 achieves the highest accuracy of 96.42%.

The paper [4] presents a smart ANPR system for contactless toll and parking management using checkpoint-based cameras instead of traditional booths. It combines OpenCV for image preprocessing, YOLOv8 for fast and accurate plate detection, and a hybrid OCR approach (Tesseract + deep learning) for character recognition. Blockchain is integrated to ensure secure, transparent, and tamper-proof payment transactions. The system achieves strong performance, with YOLOv8 reaching about 0.92–0.93 precision, 0.88–0.90 recall, and ~10–20 ms inference time. However, it remains sensitive to poor lighting, motion blur, and non-standard plate formats. The scalability of blockchain are key limitations.

From a hardware part, the solution in [5] is designed for edge deployment and is primarily implemented on embedded GPU platforms such as the NVIDIA Jetson Orin Nano, enabling on-device processing with low latency and reduced dependency on cloud computing. On the software side, it combines OpenCV for image preprocessing, YOLOv8 for real-time license plate detection, and a hybrid OCR approach (Tesseract combined with deep learning models) to improve recognition robustness under challenging conditions such as motion blur, varying illumination, and occlusions. Furthermore, blockchain technology is integrated to ensure secure, transparent, and tamper-proof transaction recording for toll and parking payments. Experimental results indicate strong performance, with YOLOv8 achieving approximately 0.92–0.93 precision, 0.88–0.90 recall, and inference times around 10–20 ms, making the system suitable for real-time edge AI applications. However, limitations remain in terms of sensitivity to non-standard license plate formats, environmental degradation of image quality, and scalability constraints of blockchain in high-throughput scenarios.

However, despite these algorithmic enhancements, the deployment environment plays a crucial role in system performance. In this context, implementing YOLO-based ANPR models on edge devices such as Raspberry Pi 5, and NVIDIA Jetson platforms allows for a practical evaluation of hardware constraints. Raspberry Pi devices provide a low-cost and energy-efficient solution, but they may face limitations in processing power when running heavier YOLO variants. In contrast, Jetson platforms, with their GPU acceleration capabilities, offer significantly better performance, enabling smoother real-time inference and reduced latency, especially for more complex models.

Beyond detection performance, the integration of blockchain technology, particularly Ethereum-based smart contracts, introduces additional considerations in terms of system efficiency and security. While Ethereum ensures tamper-proof logging, transparency, and decentralized trust, it also introduces computational overhead and transaction latency, which may vary depending on the hardware used for edge processing. Therefore, the choice of deployment platform directly impacts not only the inference speed of the YOLO model but also the responsiveness and cost efficiency of blockchain interactions. A detailed description of the proposed framework is provided in the following sections.

3. Methodology

3.1. Dataset

This study employs the License Plate Recognition Dataset (Version 11) [6] obtained from Roboflow Universe, a publicly available benchmark dataset designed for automatic license plate detection and recognition tasks. The dataset contains a total of 10,125 annotated images (510 MB) representing diverse real-world traffic environments. Images were acquired under varying operational conditions, including different illumination settings (daytime and nighttime), viewing angles, camera-to-vehicle distances, and dynamic scenarios involving both stationary and moving vehicles, as shown in Figure 1. The dataset includes heterogeneous vehicle categories, such as cars, trucks, and motorcycles, as well as regional variations in license plate designs.

All YOLO-based models were trained and evaluated under identical experimental conditions to ensure a fair and unbiased comparison. Prior to training, all images were resized to an input resolution of 416 × 416 pixels, and the corresponding license plate annotations were stored in the YOLO-compatible (.txt) format using manually annotated bounding boxes. Following the official Roboflow dataset split, the dataset was divided into 7,089 training images (70%), 2,024 validation images (20%), and 1,012 test images (10%), providing a balanced distribution for model training, hyperparameter optimization, and performance evaluation.

Training was performed with a batch size of 16 for a maximum of 150 epochs, employing an early stopping criterion with a patience of 20 epochs to mitigate overfitting. Therefore, although the maximum number of training epochs was set to 150, the actual training process terminated earlier when the early stopping criterion was met, resulting in 139, 126, and 94 training epochs for YOLOv11n, YOLOv12n, and YOLO26n, respectively. Model regularization was achieved using a dropout rate of 0.1 and a weight decay of 0.0005.

To enhance the robustness and generalization capability of the models, several data augmentation techniques were applied during training. These included Mosaic augmentation (0.5), MixUp (0.1), HSV color augmentation with hue (0.015), saturation (0.5), and value (0.3) adjustments, as well as translation (0.1), scaling (0.4), and horizontal flipping (0.5). These augmentation strategies enabled the models to better handle variations in illumination, object scale, viewpoint, and background complexity commonly encountered in real-world license plate detection scenarios.

For weight initialization, pretrained checkpoints on the COCO dataset were employed, specifically YOLOv11n, YOLOv12n and YOLO26n. The selection of these variants was guided by the constraints of the target edge deployment platform (NVIDIA Jetson Nano 2GB RAM). Larger models (e.g., YOLOv11l or YOLOv12x) were excluded due to their high memory requirements and inference latency, which are incompatible with real-time operation on embedded devices.

Figure 1. A sample picture of dataset.

3.2. Proposed method

The proposed method aims to develop a robust and efficient ANPR system by integrating 3 YOLO versions and EasyOCR/Pytesseract. This integration leverages the fast and accurate object detection capabilities of each version of YOLO and the reliable text recognition performance of EasyOCR. Additionally, the system ensures data integrity and security by registering recognized license plate numbers on the blockchain using Ethereum smart contracts. The system is designed to operate under

diverse real-world conditions, including variations in illumination, viewing angles, and image quality. The workflow of the proposed method is presented in Figure 2 and consists of the following steps:

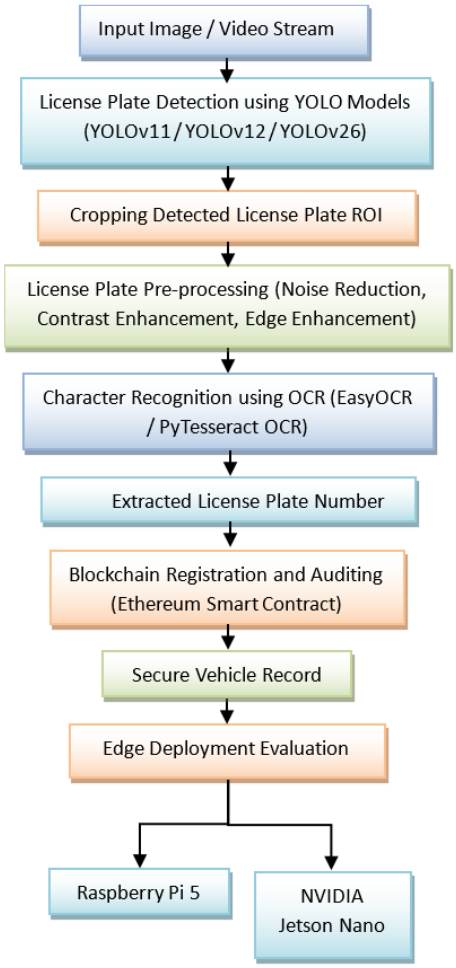

Figure 2. Image ANPR Flowchart.

1) Input: The input to the system is an image containing a vehicle with a clearly visible license plate. These images may come from surveillance cameras, mobile devices, or other imaging sources, often under challenging environmental conditions.
2) Step 1: License Plate Detection (YOLO): The first step involves detecting the license plate within the input image. The model is trained on a diverse dataset to accurately identify license plates under various conditions. As output, YOLO provides a bounding box around the detected license plate, along with a confidence score indicating the detection accuracy.
3) Step 2: Region Cropping: After detection, the bounding box coordinates are used to crop the region of interest ROI from the original image. This cropping isolates the license plate from the background, reducing noise and irrelevant information that could interfere with text recognition.
4) Step 3: Preprocessing for OCR: To enhance recognition accuracy, the cropped images undergo preprocessing before being fed into EasyOCR. Preprocessing steps include resizing to a standard size, converting to grayscale to simplify processing, and adjusting brightness and contrast to improve text visibility. Noise reduction techniques are also applied to mitigate distortions caused by environmental factors.
5) Step 4: Text Recognition (EasyOCR): The preprocessed license plate image is then passed to EasyOCR for character recognition.
6) Step 5: Validation on Raspberry Pi 5 & NVIDIA Jetson Nano: To assess the feasibility of deploying the proposed ANPR system in edge environments, the integrated YOLOv11n, YOLOv12n, YOLO26n and EasyOCR pipeline is implemented on a Raspberry Pi 5 and NVIDIA Jetson Nano device.

7) Step 6: Registering License Plate in Blockchain (Ethereum Smart Contract): Once the license plate number is recognized, the system registers it on the blockchain to ensure data integrity and immutability. Following the hybrid on-chain/off-chain schema detailed in Section 4.4, the Ethereum smart contract does not store the plain-text license plate number on-chain; instead, it securely stores a Keccak-256 cryptographic hash of the recognized license plate together with selected metadata (timestamp, access-control status, and, where applicable, parking-fee and reservation information), while the corresponding plain-text license plate number is retained only in the encrypted off-chain database. This blockchain based registration provides a tamper-proof record, enhancing the system's transparency and reliability.

The Ethereum smart contract is implemented with functions to:

1) Register a New License Plate: Store the Keccak-256 hash of the recognized plate number, together with the transaction timestamp and relevant metadata, while the plain-text plate number is kept in the encrypted off-chain database.
2) Verify Previous Registrations: Retrieve stored license plate numbers and validate their authenticity.
3) Prevent Data Manipulation: Leverage the immutable nature of blockchain to ensure that recorded data cannot be altered retroactively.

3.3. Experimental Configuration

All experiments were performed using the Ultralytics implementation (v8.4.50) of the YOLO framework on Google Colab Pro. The software environment consisted of Python 3.12.13 and PyTorch 2.10.0 with CUDA 12.8 acceleration. The OCR stage employed EasyOCR (version 1.7.2) and Tesseract OCR via the PyTesseract library (version 0.3.13), both configured for English-language license plate recognition. The OCR pipeline was implemented in Python, while image handling and plate region processing were performed using OpenCV (version 4.8.0).

To evaluate the feasibility of the deployment in edge computing environments, the trained ANPR models were deployed on a Raspberry Pi 5 (16 GB RAM) single-board computer. The device is equipped with a 64-bit quad-core ARM Cortex-A76 processor and 16 GB LPDDR4X SDRAM. And NVIDIA Jetson Nano 2GB. The device integrates a quad-core ARM Cortex-A57 64-bit CPU running at 1.43 GHz, a 128-core NVIDIA Maxwell GPU, and 2 GB LPDDR4 memory with a bandwidth of 25.6 GB/s, providing dedicated hardware acceleration for deep learning inference. The trained models were exported from the native PyTorch format (.pt) to ONNX (.onnx) using ONNX opset version 12 and executed using ONNX Runtime 1.15.1 on the Raspberry Pi 5. For the NVIDIA Jetson Nano 2 GB, the models were exported to TensorRT engine (.engine) format and executed using NVIDIA TensorRT 8.2.1 to achieve optimized inference performance.

The blockchain layer was implemented using Solidity 0.8.19, deployed and tested locally with Ganache 7.9.1 and Truffle 5.11.5. Interaction with the Ethereum network was performed via MetaMask (v11.9.1) and the Web3.py library (v6.4.0). All transaction costs were measured on a local simulated Ethereum network (Ganache) using the default gas price of 20 Gwei.

3.4. Evaluation Metrics

The principal performance evaluation metrics for training images on YOLO encompass accuracy, precision, recall and mAP@0.5. The following basic terms are defined:

1) Factual Positives (TP): A license plate is correctly detected and localized.
2) True Negatives (TN): An image or region without a license plate is correctly identified as containing no license plate.
3) False Positives (FP): A license plate is detected where none exists.
4) False Negatives (FN): A license plate present in the image is missed by the detector.

This choice was based on similar smart parking solutions ways of assessing model performance [7,8]. The following metrics were used based on the terms defined:

- Precision is determined by dividing the count of true positive predictions by the total number of positive predictions generated by the model. This metric assesses the precision of positive predictions, revealing the proportion of predicted positive instances that are genuinely relevant Equation (1).

$$Precision = \frac{TP}{TP + FP} \quad (1)$$

- Recall, often referred to as sensitivity or the true positive rate, is calculated by dividing the count of true positive predictions by the total number of actual positive instances in the dataset. This metric evaluates the model's capability to detect and identify all pertinent instances. Equation (2).

$$Recall = \frac{TP}{TP + FN} \quad (2)$$

The F1-score is defined as Equation (3).

$$F1 - score = \frac{2 \times (Precision \times Recall)}{Precision + Recall} \quad (3)$$

- For each class, the Average Precision (AP) is defined as the area under the Precision–Recall curve, obtained by plotting precision against recall as the confidence threshold is varied and computing AP as the (interpolated) integral of precision over recall, i.e., AP = $\int_0^1 P(R)\, dR$. The mean Average Precision (mAP) is then the average of the AP values over the N relevant classes, as given in Equation (4). mAP@0.5 denotes mAP computed at a single IoU threshold of 0.5, whereas mAP@0.5:0.95 denotes the mean AP averaged over IoU thresholds from 0.5 to 0.95 in steps of 0.05, following the standard COCO evaluation protocol. Higher mAP values indicate better agreement between predicted and ground-truth bounding boxes. After recognition, we measure the overall recognition rate.

$$mAP = \frac{1}{N}\sum_{c=1}^{N} AP_c \quad (4)$$

Beyond detection metrics, we further assess the recognition stage of the pipeline using:

- Plate Recognition Rate (PRR): the percentage of license plates where all characters are correctly recognized.

$$PRR = \frac{Number\ of\ correctly\ recognized\ plates}{Total\ number\ of\ plates} \times 100 \quad (5)$$

- Character Recognition Rate (CRR): the percentage of individual characters correctly recognized.

$$CRR = \frac{Number\ of\ correctly\ recognized\ characters}{Total\ number\ of character\ across\ all\ plates} \times 100 \quad (6)$$

4. Results and Discussion

4.1 Performance Comparison of YOLO Models

During the testing phase, the trained models were evaluated on unseen vehicle images using the YOLO detection framework, as illustrated in Figure 3. The objective was to accurately detect and localize license plates by predicting their corresponding bounding boxes. The models are capable of detecting one or multiple license plates within a single image. Performance was assessed using the standard object detection metrics presented in the previous section, providing a comprehensive evaluation of detection accuracy and localization performance.

Table 1 compares the performance of YOLOv11n, YOLOv12n, and YOLO26n using Precision, Recall, F1-score, mAP@0.5, mAP@0.5:0.95, inference time, training epochs, and model parameters. These metrics evaluate the trade-off between detection accuracy and computational efficiency.

Both YOLOv11n and YOLOv12n achieved the highest Precision (0.985), indicating a very low false-positive rate, while YOLOv11n obtained the highest Recall (0.945), demonstrating a slightly better ability to detect all license plates. Consequently, YOLOv11n also achieved the highest F1-score (0.9645),

reflecting the best balance between precision and recall. Although YOLO26n exhibited slightly lower detection metrics, its performance remained highly competitive. Regarding localization performance, YOLOv11n and YOLOv12n both achieved an identical mAP@0.5 of 0.963, while YOLO26n obtained 0.961. Similarly, YOLOv11n achieved the highest mAP@0.5:0.95 (0.692), followed closely by YOLOv12n (0.689) and YOLO26n (0.685), confirming the excellent localization capability of all three models.

From a computational perspective, YOLO26n is the most lightweight model, with the fewest parameters (2.50 M), the fastest inference time (1.5 ms), and the shortest training time (94 epochs). In contrast, YOLOv11n provides the highest detection performance at the cost of slightly higher computational complexity, while YOLOv12n offers performance nearly identical to YOLOv11n with similar computational requirements. Overall, **YOLOv11n** achieved the best balance between detection accuracy and localization performance, making it the most suitable model for high-reliability ANPR systems. **YOLO26n**, however, represents an attractive alternative for embedded edge devices such as NVIDIA Jetson and Raspberry Pi, where inference speed and resource efficiency are critical.

Figure 3. Examples of registration numbers.

Table 1. YOLO license plate detection performance results.

Model	Precision	Recall	F1-score	mAP@0.5	mAP@0.5:0.95	Inference Time	Epochs	parameters
YOLOv11n	**0.985**	**0.945**	**0.9645**	**0.963**	0.692	1.7ms	139	2,590,035
YOLOv12n	**0.985**	0.944	0.9640	**0.963**	**0.689**	1.9ms	126	2,568,243
YOLO26n	0.979	0.937	0.9575	0.961	0.685	**1.5ms**	94	2,504,190

4.2. OCR Recognition Performance

To evaluate the character recognition performance of the proposed ANPR system, Plate Recognition Rate (PRR) and Character Recognition Rate (CRR) were computed on the Roboflow test dataset [6]. EasyOCR, integrated into the proposed pipeline, was compared with PyTesseract using the same cropped license plate images generated by the three YOLO detection models, ensuring a fair and consistent evaluation. As shown in Table 2, EasyOCR achieved a PRR of 88.30% and a CRR of 98.54%, significantly outperforming PyTesseract, which obtained 70.32% and 86.07%, respectively. These results demonstrate the superior robustness of EasyOCR's deep learning-based architecture in accurately recognizing license plate characters under varying illumination, viewpoint, and image quality conditions. Consequently, EasyOCR was selected as the OCR engine for the proposed ANPR framework due to its higher recognition accuracy and greater reliability in real-world scenarios.

Table 2. OCR recognition results.

OCR Model	PRR(%)	CRR(%)
EasyOCR	88.30	98.54
PyTesseract	70.32	86.07

4.3 Edge deployment and Performance Evaluation

4.3.1. Experimental Setup

To assess the practical feasibility of the proposed ANPR framework in real-world intelligent transportation environments, the trained YOLO models were deployed on two representative edge-computing platforms: a Raspberry Pi 5 (16 GB RAM) and an NVIDIA Jetson Nano 2GB Developer Kit. These platforms were selected due to their widespread adoption in embedded AI applications and their contrasting hardware capabilities.

The Raspberry Pi implementation focused on lightweight CPU-based inference using both native PyTorch (.pt) and ONNX formats. In contrast, the Jetson Nano platform exploited GPU acceleration through CUDA, TensorRT, and ONNX Runtime, enabling an investigation of the accuracy-performance trade-off under resource-constrained edge conditions.

For all experiments, detection performance was evaluated using Precision, Recall, and mAP@0.5. To assess deployment efficiency on embedded platforms, the inference latency of each model was measured on the Raspberry Pi 5 and NVIDIA Jetson Nano using the corresponding PyTorch, ONNX Runtime, and TensorRT implementations.

4.3.2. Performance Evaluation on Edge Platforms

Table 3 compares the detection performance of YOLOv11n, YOLOv12n, and YOLO26n on Raspberry Pi and NVIDIA Jetson Nano using different deployment formats, including PyTorch (.pt), ONNX, and TensorRT engines. Overall, the results indicate that model conversion and hardware acceleration significantly influence both detection accuracy and inference efficiency.

Table 3. Embedded detection performance of YOLOv11n, YOLOv12n, and YOLO26n on Raspberry Pi 5 and NVIDIA Jetson Nano.

Metric	**precision**				**recall**				**mAP@0.5**			
	Raspberry Pi		**NVIDIA Jetson**		**Raspberry Pi**		**NVIDIA Jetson**		**Raspberry Pi**		**NVIDIA Jetson**	
Model	**PyTorch**	**ONNX**	**PyTorch**	**TensorRT**	**PyTorch**	**ONNX**	**PyTorch**	**TensorRT**	**PyTorch**	**ONNX**	**PyTorch**	**TensorRT**
YOLOv11n	0.969	0.946	0.942	0.924	0.671	0.702	0.697	0.771	0.741	0.746	0.741	0.828
YOLOv12n	0.939	0.916	0.962	0.940	0.745	0.700	0.723	0.681	0.780	0.753	0.753	0.737
YOLO26n	0.922	0.911	0.831	0.911	0.750	0.657	0.723	0.657	0.752	0.703	0.748	0.703
(PyTorch / TensorRT on Jetson)												

On the Raspberry Pi platform, (Figure 4) YOLOv11n achieved the highest precision in both PyTorch (0.969) and ONNX (0.946) formats, indicating a lower false-positive rate. In contrast, YOLOv12n obtained the highest recall (0.745) and mAP@0.5 (0.780), demonstrating superior localization performance and improved capability to detect license plates under diverse conditions. YOLO26n provided competitive results, achieving the highest recall among some configurations while maintaining a satisfactory mAP@0.5.

Figure 4. ANPR on Raspberry Pi.

On the Jetson Nano platform, TensorRT optimization improved the detection performance of YOLOv11n, increasing recall from 0.697 to 0.771 and mAP@0.5 from 0.741 to 0.828. This result suggests that hardware-specific optimization can enhance both localization accuracy and detection robustness. Conversely, YOLOv12n experienced a slight reduction in recall and mAP after TensorRT conversion, highlighting that optimization techniques may affect models differently depending on their architecture. YOLO26n maintained competitive precision values but exhibited lower recall and mAP compared with YOLOv11n.

Table 4 presents the corresponding inference latency measurements. On the Raspberry Pi, converting the models from PyTorch to ONNX substantially reduced inference time. For example, YOLOv11n latency decreased from 98.2 ms (10 FPS) to 38.3 ms (26 FPS), while YOLOv12n improved from 111.0 ms (9 FPS) to 40.7 ms (25 FPS). The fastest Raspberry Pi deployment was achieved by YOLO26n ONNX, requiring only 33.4 ms per image (30 FPS). It should be emphasized that this figure represents the YOLO detector's forward-pass inference time only; it does not include OCR, pre/post-processing, database operations, or blockchain communication latency, and should therefore be interpreted as an indicator of detector-level real-time capability rather than complete end-to-end ANPR system throughput (see Section 4.3.1). A similar trend was observed on the NVIDIA Jetson platform. TensorRT optimization reduced the inference time of YOLOv11n from 52.0 ms (19 FPS) to 38.9 ms (26 FPS), representing an improvement of approximately 25%. For YOLOv12n, TensorRT reduced latency from 127.9 ms (8 FPS) to 88.2 ms (11 FPS), corresponding to a performance gain of nearly 31%. YOLO26n was evaluated on PyTorch and ONNX only, with inference time decreasing from 187.4 ms (5 FPS) to 137.6 ms (7 FPS), representing a performance gain of approximately 26.6%. Although ONNX significantly accelerates YOLO26n on the Jetson platform, its inference speed remains below that achieved by TensorRT-optimized YOLOv11n and YOLOv12n. Among all evaluated Jetson configurations, YOLOv11n TensorRT achieved the lowest inference latency (38.9 ms, 26 FPS), followed by YOLOv12n TensorRT (88.2 ms, 11 FPS). These results indicate that only the YOLOv11n TensorRT configuration reaches real-time performance on the Jetson platform, whereas the remaining configurations provide lower throughput. As with the Raspberry Pi experiments, these FPS values correspond exclusively to detector inference and do not represent the complete ANPR processing pipeline.

Table 4. Inference time and throughput (FPS) of YOLO models on Raspberry Pi 5 and NVIDIA Jetson Nano using PyTorch, ONNX, and TensorRT.

Metric	**Inference time(ms)**			
	Raspberry Pi		**NVIDIA Jetson**	
Model	**PyTorch**	**ONNX**	**Pytorch**	**TensorRT**
YOLOv11n	98.2 (10 FPS)	38.3 (26 FPS)	52.0 (19 FPS)	38.9 (26 FPS)
YOLOv12n	111.0 (9 FPS)	40.7 (25 FPS)	127.9 (8 FPS)	88.2 (11 FPS)
YOLO26n *(PyTorch / ONNX on Jetson)*	89.8 (11 FPS)	33.4 (30 FPS)	187.4 (5 FPS)	137.6 (7 FPS)

Frames-per-second (FPS) values shown in parentheses were computed as FPS = 1000/latency (ms) for each reported inference time.

Overall, the results reveal a clear trade-off between detection accuracy and computational efficiency, and no single model is uniformly superior across all deployment configurations. YOLOv12n attains the highest mAP@0.5 on the Raspberry Pi (PyTorch) and the highest precision on the Jetson platform (PyTorch), whereas YOLOv11n achieves the best overall balance between detection performance and computational efficiency. In particular, TensorRT optimization enables YOLOv11n to reach 38.9 ms (26 FPS) on the Jetson platform while also providing the highest TensorRT mAP@0.5. Meanwhile, YOLO26n demonstrates the fastest inference on the Raspberry Pi when exported to ONNX (33.4 ms, 30 FPS), making it an attractive candidate for highly resource-constrained edge deployments where inference speed is prioritized over absolute detection performance. These findings confirm that model optimization through ONNX and TensorRT is essential for deploying ANPR systems on embedded platforms, and that the preferred model depends on the target hardware and optimization framework rather than being fixed across all deployment scenarios.

4.4. Implementation of Blockchain Technology

To ensure the secure management of data related to vehicle movements, particularly for vehicles entering and exiting access-controlled parking facilities, we have implemented a blockchain-based backup solution. Blockchain technology offers a decentralized and tamper-resistant storage mechanism, providing integrity, immutability, and traceability of the recorded information; confidentiality of

sensitive data is achieved separately through the encrypted off-chain database described in Section 4.4, since on-chain data is, by design, visible to all network participants. In this context, Ethereum was selected due to its reliability and support for smart contract functionality. It enables the generation of verifiable logs for each vehicle entry and exit event, ensuring traceability while preventing any unauthorized modification or data manipulation.

It should be emphasized that the proposed blockchain implementation represents a proof-of-concept evaluated on a local Ethereum network using Ganache. The primary objective is to validate the smart contract logic, transaction workflow, and gas consumption under controlled conditions. Consequently, the current implementation should not be interpreted as a production deployment on the public Ethereum network or on a permissioned blockchain infrastructure.

This approach is particularly important for high-security access-control applications, where data sensitivity and security requirements are critical. By leveraging the Ethereum blockchain, each transaction is securely timestamped and cryptographically protected, providing transaction authentication, integrity, and traceability through digital signatures; as noted above, confidentiality of the underlying license-plate data is provided by the off-chain encryption layer rather than by the blockchain itself. Furthermore, the decentralized architecture eliminates the need for a central authority for validation, thereby reducing the risk of single points of failure. Overall, this solution enhances system security and provides a trustworthy and persistent audit trail for future verification of vehicle access events, while remaining resilient to cyber threats. (Figure 5)

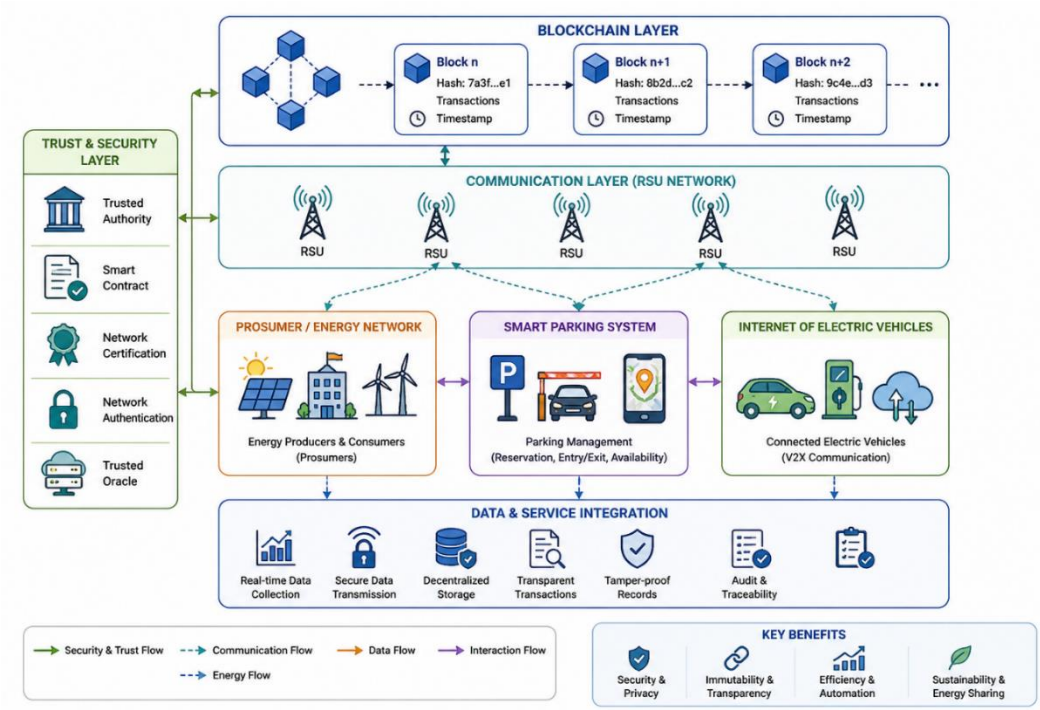

Figure 5. Smart parking model for Smart City.

In our implementation of the blockchain-based parking system, we utilize a local development setup by replacing the main Ethereum network with a more practical testing environment based on Ganache and MetaMask. Ganache acts as a personal Ethereum blockchain that facilitates the development, testing, and deployment of smart contracts in a controlled and simulated environment. This allows developers to execute and evaluate smart contract functionalities efficiently without incurring real transaction fees.

MetaMask, on the other hand, is used as a browser-based wallet and gateway to the blockchain. It enables users to securely manage Ethereum accounts, authorize transactions, and interact with decentralized applications. When operating with Ganache, the system automatically generates multiple accounts, each associated with a unique pair of public and private keys. Every account is initialized with a predefined balance of 100 ETH, where Ether represents the native cryptocurrency used for transaction simulation within the Ethereum ecosystem.

To minimize blockchain storage overhead while preserving data integrity, the proposed architecture

adopts a hybrid on-chain/off-chain storage strategy. Only lightweight metadata are recorded on-chain, including a cryptographic hash of the license plate identifier, transaction timestamp, access control status, parking fee, vehicle integrity status, and parking slot reservation information. Sensitive information is securely maintained in an encrypted off-chain database.

Before being submitted to the blockchain, each recognized license plate is transformed using the Ethereum Keccak-256 cryptographic hash function. This irreversible cryptographic digest enables integrity verification without exposing the original license plate number on-chain. During verification, the newly recognized license plate is hashed again and compared with the immutable hash stored on the blockchain.

Each authorized entity interacting with the blockchain owns an Ethereum account associated with a unique public/private key pair. The private key is used to digitally sign blockchain transactions before submission, while the corresponding public key enables verification of transaction authenticity and integrity, as described in our previous works [9,10].

Table 5 presents the execution cost of the proposed smart contracts. The reported transaction costs were obtained on a local Ganache blockchain, where the gas price was fixed and deterministic. It is important to note that Ganache is a simulated, local Ethereum environment: it does not broadcast transactions to a real network and therefore does not incur any actual monetary transaction fees. For comparison purposes, the ETH-to-USD conversion was performed using a constant reference value of 1 ETH = 3300 USD (July 2025). Therefore, the reported USD figures represent normalized mainnet-equivalent cost estimates. These were calculated by applying a fixed gas price and ETH/USD rate to the gas consumption measured on Ganache, rather than actual financial costs incurred. On the public Ethereum network, gas prices fluctuate dynamically, so actual costs would differ from these estimates. The following comparison of Hyperledger Fabric, Bitcoin, and Ethereum is conceptual, based on the platforms' published design characteristics, and does not reflect an experimental evaluation performed in this study. Hyperledger Fabric and Bitcoin represent two very different blockchain paradigms, and Ethereum can be seen as an intermediary solution that bridges the gap between them. On one hand, Hyperledger Fabric is a permissioned blockchain designed for enterprise environments, offering strong performance and controlled access, but with relatively limited interoperability and flexibility. On the other hand, Bitcoin is primarily focused on peer-to-peer financial transactions and relies on a constrained scripting language that does not support advanced application logic.

Ethereum distinguishes itself by enabling full smart contract execution through the Ethereum Virtual Machine, which allows developers to build complex and automated decentralized applications. This programmability makes it significantly more expressive than Bitcoin while remaining more open and flexible than typical permissioned systems.

Another key advantage of Ethereum lies in its adaptability to different deployment contexts. It supports both public and private network configurations, such as Proof of Authority setups, while maintaining compatibility of smart contracts across environments without major modifications.

Table 5. Smart contract Transaction cost.

Smart Contract	Transaction Cost ($\times 10^{-3}$ ETH)	Cost (USD)	Gas Units*
Access Control Smart Contract	0.5	1.65	25,000
Parking Fee Smart Contract	0.8	2.64	40,000
Vehicle Status and Data Integrity Smart Contract	0.4	1.32	20,000
Parking Slot Reservation Smart Contract	1.0	3.30	50,000

* Gas units were derived from the reported ETH cost and the fixed Ganache gas price of 20 Gwei (1 Gwei = 10^{-9} ETH) stated in Section 3.3, i.e., Gas units = ETH cost / 20×10^{-9}. Each value in this table corresponds to the execution cost of a single invocation of the corresponding smart-contract function (not contract deployment, and not the aggregated cost of a full vehicle event, which may invoke more than one contract; see the discussion in Section 5).

In terms of scalability, the current prototype uses a local Ganache testnet that simulates Ethereum's execution environment without replicating mainnet congestion. The four smart contracts deployed in this study generate between 0.4×10^{-3} and 1.0×10^{-3} ETH per transaction (Fig 6), corresponding to 1.32–3.30 USD per transaction at current rates.

5. Discussion

The experiments highlight distinct deployment characteristics for the two edge platforms. Raspberry Pi 5 achieved substantially lower inference latency, particularly when using ONNX models, reaching

near real-time performance with inference times below 40 ms. In contrast, the Jetson Nano benefited from GPU acceleration but remained constrained by its limited 2 GB memory capacity, resulting in extensive swap usage and lower throughput.

From the training evaluation (Table 1), YOLOv11n achieved the best overall detection performance, providing the highest recall, F1-score, and mAP@0.5:0.95, while YOLOv11n and YOLOv12n achieved identical precision and mAP@0.5. During deployment, the optimized models exhibited different behavior depending on the target hardware and optimization format. On the NVIDIA Jetson platform, YOLOv12n achieved the highest precision in its native PyTorch implementation (0.962), whereas YOLOv11n delivered the best overall deployment performance after TensorRT optimization, achieving the highest recall (0.771), the highest mAP@0.5 (0.828), and the lowest inference latency (38.9 ms, 26 FPS). Although YOLO26n benefited from ONNX optimization, reducing its inference time from 187.4 ms to 137.6 ms (approximately 26.6% improvement), it remained slower than the TensorRT-optimized YOLOv11n and YOLOv12n configurations.

This work is limited to evaluating transaction costs on a local Ganache blockchain. Raspberry Pi is used here solely as an embedded hardware host for the ANPR pipeline and is not a blockchain consensus mechanism; no proof-of-work, proof-of-stake, or other consensus protocol was implemented or simulated on this hardware. Ganache does not replicate the consensus, congestion, or confirmation-delay behavior of a public Ethereum network because it does not use a true consensus algorithm. Instead, it instantly mines each submitted transaction into its own block. Regarding the issue of privacy and security, our previous work has addressed this point [11, 12]. The smart contract leverages this relationship to authenticate the user's identity and authorize actions. To scrutinize the certificate for tampering or fabrication, the system verifies the digital signature using the corresponding public key and compares it against the recorded certificate data.

Beyond cost evaluation, blockchain adoption in Intelligent Transportation Systems must be analyzed in terms of scalability, privacy, and consensus performance. As highlighted in [13], blockchain has already demonstrated strong potential in securing sensitive IoT-based ecosystems such as the Internet of Medical Things (IoMT). The same principles apply to vehicular data in ANPR systems, where privacy-preserving mechanisms and encrypted communication are essential to protect user identities and location data. Leveraging blockchain's decentralization and immutability ensures that license plate transactions are tamper-proof, while compliance with data protection regulations is maintained. This comparison underlines the broader applicability of blockchain security models beyond healthcare and into transportation domains.

Another important dimension relates to blockchain consensus. As [14] emphasizes, consensus protocols such as Proof of Stake, Proof of Work, and Proof of Authority each introduce trade-offs between security, latency, and energy efficiency. Hardware-accelerated implementations, particularly FPGA-based designs, have been shown to improve execution time and throughput, directly addressing one of the main bottlenecks in deploying blockchain-secured systems at scale. For ANPR applications, where transaction validation and event logging must occur in near real-time, efficient consensus mechanisms are crucial. The integration of lightweight or hardware-optimized consensus strategies can therefore enhance both scalability and responsiveness of parking systems.

For a medium-scale parking facility processing approximately 500 vehicle events per day, and assuming a single smart-contract invocation per event, Table 4 indicates a daily blockchain cost ranging from 500 × 1.32 = 660 USD/day (Vehicle Status and Data Integrity contract, the cheapest) to 500 × 3.30 = 1,650 USD/day (Parking Slot Reservation contract, the most expensive). If a complete vehicle event instead requires invoking all four deployed smart contracts (Access Control, Parking Fee, Vehicle Status and Data Integrity, and Parking Slot Reservation), the combined per-event cost is 1.65 + 2.64 + 1.32 + 3.30 = 8.91 USD/event, corresponding to 500 × 8.91 = 4,455 USD/day. The exact number of contracts invoked per event depends on the transaction workflow: in the current proof-of-concept, a routine entry/exit event invokes the Access Control and Vehicle Status/Data Integrity contracts, while the Parking Fee and Parking Slot Reservation contracts are invoked only for events involving paid parking or a prior reservation, respectively; consequently, the realistic per-event cost lies between the single-contract and four-contract bounds reported above depending on event type. For large-scale ITS deployments (>10,000 events/day), deploying on a private Ethereum PoA network (e.g., Geth Clique) would reduce costs to negligible levels while preserving smart contract functionality and immutability within a controlled consortium. Mainnet deployment without Layer-2 scaling solutions (e.g., Optimism, Arbitrum) would be cost-prohibitive at scale and is not recommended for production use. These large-scale projections are cost estimates based on published gas-price assumptions rather than results of an

experimental evaluation; transaction throughput, confirmation latency, network congestion, and blockchain storage growth at these scales were not measured in this work and remain directions for future research (Section 6).

In broad terms, our work contributes primarily at the application level, demonstrating blockchain based automation for parking management. However, the integration of advanced security frame works from IoT domains [13] and optimized consensus protocols [14] would further strengthen the proposed architecture. These insights highlight future research opportunities to combine privacy preserving blockchain mechanisms with hardware-accelerated consensus, paving the way for secure, scalable, and efficient deployment of blockchain-secured ANPR systems in real-world intelligent transportation environments.

To provide insight into system limitations, we analyzed false positive and false negative detection cases from the test set. False negatives (missed detections) occurred primarily under three conditions:

(1) Extreme low-angle captures (>45° horizontal tilt), where the plate aspect ratio deviates substantially from training distribution;
(2) Severe motion blur in images of moving vehicles, particularly at shutter speeds below 1/100s equivalent; and
(3) Partial occlusion by dirt, stickers, or physical damage covering >30% of the plate area.

For OCR failures, the dominant error sources were:

(1) Low-contrast plate-character combinations (e.g., white characters on silver plates under overexposed conditions);
(2) Non-standard fonts on custom or vintage plates; and
(3) Character-level ambiguities ('0'/'O', '1'/'I', '8'/'B') under degraded image quality, where EasyOCR's confidence score fell below 0.6.

Although the proposed framework demonstrates the feasibility of integrating blockchain with ANPR, several practical aspects remain to be investigated. In particular, transaction throughput, confirmation latency, blockchain storage growth, network congestion, and the capability of handling high-frequency ANPR events have not been evaluated in this work. Furthermore, comparing Ethereum with permissioned blockchain frameworks such as Hyperledger Fabric would provide valuable insights into scalability, performance, and deployment trade-offs. These aspects constitute important directions for future research.

6. Conclusion & Future Work

This work introduces a blockchain-enabled Automatic Number Plate Recognition (ANPR) framework with multiple potential applications, demonstrating its suitability for accurate and real-time vehicle identification on resource-constrained edge platforms. The proposed system was comprehensively evaluated across all stages of the recognition pipeline, including license plate detection, optical character recognition, blockchain-based data storage, and embedded deployment, confirming its effectiveness for intelligent transportation applications.

Among the evaluated detection models, YOLOv11n provided the best overall detection performance during training, achieving the highest recall, F1-score, and mAP@0.5:0.95, while YOLOv11n and YOLOv12n achieved identical precision and mAP@0.5. In contrast, YOLO26n demonstrated the highest computational efficiency on the Raspberry Pi when deployed using ONNX, achieving the lowest inference latency. However, on the NVIDIA Jetson platform, YOLOv11n optimized with TensorRT provided the best balance between detection performance and inference speed, achieving the highest recall and mAP@0.5 while also delivering the lowest inference latency (38.9 ms, 26 FPS). These results demonstrate that hardware-specific optimizations, including ONNX and TensorRT, can significantly alter the trade-off between accuracy and inference speed, highlighting the importance of evaluating object detection models directly on the target edge platform.

Nevertheless, the performance of deep learning-based computer vision systems strongly depends on the availability of large-scale, diverse, and well-annotated datasets. In this study, 10,125 annotated images from the Roboflow repository were employed to train and evaluate the proposed framework. Although the experimental results demonstrate promising detection and recognition performance, the limited diversity of publicly available datasets remains a challenge for achieving robust generalization under highly variable real-world conditions.

Future work will focus on developing a larger and more comprehensive vehicle image dataset that

includes challenging scenarios such as low illumination, occlusions, motion blur, adverse weather conditions, varying camera viewpoints, and irregular license plate formats. Such a dataset will improve the robustness, generalization capability, and benchmarking reliability of the proposed ANPR framework.

From the blockchain perspective, future research will investigate comprehensive scalability analyses, including transaction throughput, confirmation latency, blockchain storage growth, dynamic gas pricing, and energy consumption. Comparative studies between public blockchain platforms (e.g., Ethereum) and permissioned blockchain frameworks (e.g., Hyperledger Fabric) will also be conducted to identify the most suitable architecture for real-world intelligent transportation systems.

To further strengthen system security and privacy, advanced cryptographic mechanisms, including zero-knowledge proofs and homomorphic encryption, will be integrated to mitigate identity leakage and unauthorized access while preserving the transparency and integrity of blockchain transactions. In parallel, lightweight and hardware-accelerated consensus mechanisms will be explored to reduce computational overhead and improve scalability for real-time edge deployments.

In addition, interoperability between IoT devices and blockchain infrastructures will be investigated, with particular emphasis on secure cross-domain communication and compliance with emerging data protection regulations. The proposed framework will also be evaluated on heterogeneous edge computing platforms to analyze the trade-offs among inference latency, computational efficiency, energy consumption, and deployment cost under realistic intelligent transportation scenarios.

Finally, future work will include a comprehensive profiling study comparing execution time, resource utilization, and energy consumption across different hardware environments, including Raspberry Pi, NVIDIA Jetson platforms, and GPU-based systems. Detailed analyses of inference latency, CPU/GPU utilization, memory consumption, thermal behavior, and computational efficiency will be reported. Moreover, dedicated security and performance evaluation tools, such as Scyther and Hyperledger Caliper, will be employed to conduct an in-depth assessment of the robustness, scalability, and deployment readiness of the proposed blockchain-enabled ANPR framework.

Author Contributions

Conceptualization, N. A. and J.K.; methodology, N.A., J.K. ; software, N.A.; validation, J.K., N.A. and H.H.; formal analysis, N.A.; investigation, J.K. and N.A.; resources, J.K. and H.H.; data curation, N.A.; writing—original draft preparation, N.A.; writing—review and editing, J.K., N.A. and H.H.; visualization, N.A.; supervision, J.K. and H.H.; project administration, J.K. and H.H. All authors have read and agreed to the published version of the manuscript.

Funding

This research received no external funding

Conflicts of Interest

The authors declare that they have no conflict of interest.

Data Availability

There is no data availability.

References

1 N. Affes, J. Ktari, N. Ben Amor, T. Frikha, and H. Hamam. “Real-Time Detection and Tracking in Multi-Speaker Video Conferencing”. In *Intelligent Systems Design and Applications (ISDA 2022)*, Lecture Notes in Networks and Systems, Vol. 716, Springer, Cham, pp. 119–130, 2023. DOI: 10.1007/978-3-031-35501-1_11.

2 P. Palsodkar, A. Balpande, A. Hedaoo, K. Atara, H. Madankar, and S. Deshmukh. “Edge AI-Based Real-Time Automatic License Plate Recognition Using YOLOv8 on NVIDIA Jetson Orin Nano”. In *Proceedings of the 3rd International Conference on Emerging Trends in Engineering and Medical Sciences (ICETEMS)*, IEEE, pp. 1–6, 2026.

3 R. R. Akurathi, S. Dasam, and A. V. Pushpavathi. “Enhancing Road Safety and Compliance: A Reliable Number Plate Recognition System on Raspberry Pi with Integrated Rear-View Camera Analysis”. In *Proceedings of the 4th International Conference on Innovative Sustainable Computational Technologies (CISCT)*, IEEE, pp. 1–6, 2024.

4 M. N. Samarth, S. Gopisetti, and M. Chaitra. “Smart ANPR: A Checkpoint-Based Toll and Parking Management System Using Hybrid OCR and Blockchain”. In *Proceedings of the International Conference on Intelligent and Innovative Technologies in Computing, Electrical and Electronics (IITCEE)*, IEEE, pp. 1–7, 2025.

5 F. S. Jamali and M. Sadedel. “An Embedded Real-Time Automatic License Plate Recognition System Using YOLO Algorithm”. *Scientia Iranica*, 2025.

6 Roboflow Universe. “License Plate Recognition Dataset (Version 11)”. Available at: https://universe.roboflow.com/roboflow-universe-projects/license-plate-recognition-rxg4e/dataset/11 (Accessed on date: 28 July 2026)

7 H. Bura, N. Lin, N. Kumar, S. Malekar, S. Nagaraj, and K. Liu. “An Edge-Based Smart Parking Solution Using Camera Networks and Deep Learning”. In *Proceedings of the IEEE International Conference on Cognitive Computing (ICCC)*, IEEE, pp. 17–24, 2018.

8 P. Chantri, S. Chaiyaboon, P. Prathan, and T. Sirapob. “A Camera-Based Smart Parking System Employing Low-Complexity Deep Learning for Outdoor Environments”. In *Proceedings of the 17th International Conference on ICT and Knowledge Engineering (ICTKE)*, IEEE, pp. 1–5, 2019.

9 J. Ktari, T. Frikha, M. A. Yousfi, M. K. Belghith, and N. Sanei. “Embedded Keccak Implementation on FPGA”. In *Proceedings of the IEEE International Conference on Design & Test of Integrated Micro & Nano-Systems (DTS)*, Cairo, Egypt, IEEE, pp. 1–5, 2022. DOI: 10.1109/DTS55284.2022.9809847.

10 J. Ktari, T. Frikha, N. Ben Amor, L. Louraidh, H. Elmannai, and M. Hamdi. “IoMT-Based Platform for E-Health Monitoring Based on the Blockchain”. *Electronics*, Vol. 11, No. 15, Art. 2314, 2022.

11 B. L. Sahu and P. Chandrakar. “Blockchain-Oriented Secure Communication and Smart Parking Model for Internet of Electric Vehicles in Smart Cities”. *Peer-to-Peer Networking and Applications*, Vol. 18, 2025.

12 S. A. B. Mohamed, A. Talha, J. Ktari, Y. B. Taleb, T. Frikha, and H. Hamam. “Blockchain-Powered Smart System Platform Development: Use Case–Water Meter Readings.” *Journal of Information Assurance and Security*, *19*(5), 214–232, 2025.

13 Y. Y. Ghadi, T. Mazhar, T. Shahzad, M. A. Khan, A. Abd-Alrazaq, A. Ahmed, and H. Hamam. “The Role of Blockchain to Secure Internet of Medical Things”. *Scientific Reports*, Vol. 14, No. 1, Art. 18422, 2024.

14 J. Ktari, T. Frikha, M. Hamdi, and H. Hamam. “Enhancing Blockchain Consensus with FPGA: Accelerating Implementation for Efficiency”. *IEEE Access*, Vol. 12, pp. 44773–44785, 2024.