Journal Home Page: https://sjes.univsul.edu.iq/

Research Article:

Intelligent License Plate Recognition System for Smart Campus Access Control

Bnar Abdulsalam Abdulrahman [1,a,*]
Mohammed Abdulmaged Faraj [2,a]
[a] Computer Science Department , Komar University of Science and Technology ,KR, Iraq

Article Information

Article History:
Received: June 15, 2026
Accepted: August 3, 2026
Available online: August , 2026

Keywords:
License Plate Recognition, YOLOv8, EasyOCR, Smart Campus, Access Control, Computer Vision, Deep Learning, Django, Intelligent Transportation Systems.

About the Authors:

Corresponding author:
Ms.Bnar Abdulsalam Abdulrahman
E-mail: bnar.abdoulsalam@komar.edu.iq
Researcher Involved:
Mr.Mohammed Abdulmaged Faraj
E-mail: mohammed.abdulmaged@komar.edu.iq

DOI https://doi.org/10.17656/sjes.10225

Abstract

Modern academic campuses face a growing tension between convenient vehicle access and increasing demands for security and operational efficiency. At scale, manual gate checks are slow, error-prone, and difficult to audit. This paper presents the design, implementation, and evaluation of an intelligent License Plate Recognition (LPR) system for smart-campus access control. The system integrates a fine-tuned YOLOv8 detector for vehicle and plate localization, the EasyOCR engine for real-time plate-text recognition, a MySQL relational database that matches recognized plates against authorized and blacklisted vehicle lists, and a Django web dashboard that provides administrative visibility. A blacklist-aware alerting subsystem dispatches SMS and e-mail notifications whenever an unauthorized or blacklisted vehicle is detected. The novelty of this work lies not in any single algorithmic component but in the complete, reproducible, open-source integration of detection, recognition, database matching, logging, and alerting into one deployable framework adapted to local plate formats and campus operational requirements. The detector was trained on a combination of public benchmarks and a locally collected campus dataset and evaluated on a held-out test set of 850 images, achieving a plate-detection mAP@0.5 of 0.942, precision of 0.945, recall of 0.938, full-plate OCR accuracy of 92.1%, end-to-end recognition accuracy of 94.8%, and an average throughput of 23.5 FPS on commodity hardware. The architecture is modular, scalable, and deployable on Commercial Off-The-Shelf (COTS) hardware, offering an affordable and replicable solution for budget-constrained institutions.

1. Introduction

University, research institute, and larger business facilities tend to have much higher daily traffic from vehicles. On the other hand, conventional physical access control (manual gate inspection, paper-based logs or RFID stickers) has several well-documented drawbacks: it is labour-intensive and prone to human error, it can be circumvented via unmanned gates or fences, and provides little analytical detail on traffic patterns (Badawy et al., 2019; Swiftlane, 2024a). With the size and complexity of campuses around the world continuing to rapidly grow, there is an increasing demand for intelligent, automated approaches to access control, especially those that are designed to meet stringent auditability requirements. At the same time, surveillance cameras have become increasingly affordable, and powerful edge-computing devices optimized for deep-learning inference make it possible not only to monitor traffic and parking availability but also to inspect each vehicle that passes through the field of view. Because of the breakthroughs in computer vision and deep learning, License Plate Recognition (LPR) systems are becoming feasible to use on live video streams. An LPR system, built on accurate object detection and robust Optical Character Recognition (OCR), can monitor traffic, verify whether vehicles are authorized, and log every entering vehicle without human intervention (Du et al., 2013). Modern single-stage detectors have reached high accuracy and frame

rates suitable for real-time monitoring (Redmon et al., 2016), while existing OCR engines such as EasyOCR can identify license plates in a variety of fonts, sizes, and lighting conditions (JaidedAI, 2020). This has advanced integrated security systems in the wider smart-campus market where IoT-enabled campuses are progressively integrating sensor networks, near-real-time analytics and automated response systems to minimise operational cost while maximising safety (Hashstudioz, 2024; rf IDEAS, 2024). LPR is a valuable sensing modality that turns low-level, unstructured video into high-level campus intelligence. However, although individual sub-tasks (detection benchmarks, OCR datasets) are well studied, fully working, integrated, end-to-end LPR systems deployed in real institutional settings remain comparatively rare in the published literature.

Driven by the operational requirements of a campus environment, this paper presents the design, implementation, and evaluation of an end-to-end intelligent LPR system. Unlike most studies that focus on a single sub-task, we deliver the full pipeline: system architecture, dataset preparation, model-selection justification, image preprocessing, database and web-dashboard design, and an alerting workflow. Every design decision is grounded in up-to-date published evidence and assessed against real-world deployment factors.

1.1 Problem Statement

The majority of current campus gates rely on human operators and simple physical credentials such as sticker-based RFID tags or ID cards. These traditional methods have structural weaknesses that an LPR-based approach directly addresses: (i) they do not scale: every vehicle must carry a pre-issued credential, so visitor and contractor vehicles cannot be handled automatically, whereas an LPR system identifies any vehicle by its plate; (ii) credentials can be lost, forgotten, lent, or cloned, while a license plate is a legally mandated, externally visible identifier; (iii) manual checks and RFID logs produce little structured data that can be audited or analyzed, whereas an LPR system automatically generates a searchable, time-stamped, image-backed audit trail; and (iv) guards and RFID readers cannot proactively flag blacklisted or suspicious vehicles, while an LPR system can raise an alert the moment such a plate appears. Moreover, commercial LPR products are typically cost-prohibitive for budget-limited educational institutions because they are proprietary and often coupled to specific hardware (Campus Safety Magazine, 2024). The absence of a cost-effective, easily replicable, open-source LPR solution tailored to a campus environment is the gap this work addresses.

1.2 Research Objectives

The main objectives of this research are:

- Build a License Plate Detection Project in Real Time with YOLOv8 deep learning architecture.
- Reliable Text extraction of license plates using OCR with EasyOCR using preprocessing adapted to handle various environmental conditions.
- To match the detected plates with a permitted vehicle database to classify entry events as authorized, unauthorized or blacklisted.
- Automatically log entry time, status and a snapshot as evidence in a structured relational database of the vehicle.
- To implement a search based web interface for campus administrators to monitor and audit access events.
- To send real-time SMS and email notifications when flagged or unwanted vehicles are recorded.
- To provide AI-enabled continuous and auditable monitoring of campus security.

1.3 Contributions

To state the novelty of this work explicitly: the individual components (YOLOv8, EasyOCR, MySQL, Django, Twilio) are established technologies, and we do not claim algorithmic novelty in any one of them. The contribution is at the system-integration and deployment-engineering level, which the literature review (Section 2.7) shows to be under-represented in published work. Specifically, this paper makes four contributions. First, it introduces a complete and reproducible system architecture that integrates detection, recognition, database matching, logging, and alerting into a single framework driven by campus authorization workflows, an integrated design rarely documented end-to-end in prior literature. Second, the entire system is built from open-source technologies (YOLOv8, OpenCV, EasyOCR, MySQL, and Django), so it can be adopted by other institutions without licensing fees, and it is described in sufficient detail to be replicated directly. Third, the paper documents the engineering decisions (image preprocessing, plate-format validation, blacklist-first matching logic, deduplication, database schema and indexing, and alert-escalation policies) that determine whether an LPR system works reliably on a real campus rather than in a controlled laboratory setting, including the adaptation of the detector and OCR stage to local plate formats through fine-tuning on a locally collected campus dataset. Fourth, the system is validated in an empirical evaluation, with quantitative detection, recognition and end-to-end throughput results in Section 5 and a comparison to recent work. So in terms of the classic contribution categories, this contribution would effectively be system integration and deployment methodology with AI-model adaptation (fine-tuning) for the local environment; custom hardware is deliberately

avoided, as the design goal is to rely on low-cost Commercial Off-The-Shelf components.

2. Literature Review

2.1 Early and Classical Approaches to License Plate Recognition

LPR has been an active research area for over twenty years. Many of the first pipelines were based on classical computer-vision methods such as edge detection and morphological operations, histogram analysis or template matching (Du et al., 2013). These techniques performed well only under comparatively rigid conditions with static camera positions, uniform lighting and consistent plate size; performance degraded under variable illumination, variable camera angles, plate tilt and occlusion. Fu (2024) surveyed the shift from classical Automatic License Plate Recognition (ALPR) methods, and their failure modes, to learned representations. The classical pipeline consists of three serial blocks (license plate detection, character segmentation and individual character recognition), where each stage is a separate point of failure: if the first stage fails, everything downstream cannot recover.

2.2 Deep Learning-Based Detection

Over the course of deep learning, detection and recognition were primarily solved by convolutional neural networks (CNNs). Among region-based detectors, Faster R-CNN (Ren et al., 2015), a two-stage detector that uses region proposals for accurate localization, was one of the highest regarded approaches to date but too expensive for online inference on commodity hardware. Single-stage detectors attempted to overcome this limitation: YOLO (Redmon et al., 2016) yields bounding boxes and class scores with a single forward pass, allowing real-time inference on commodity GPUs. Subsequent YOLO versions have steadily improved both accuracy and speed. YOLOv8, released in January 2023 (Jocher et al., 2023), features an anchor-free detection head and modifies the C2f backbone module for more effective multi-scale feature aggregation with real-time throughput and higher mean Average Precision (mAP) on the COCO benchmark.

The most recent studies show that YOLOv8 models have better performance than older YOLO models in LPR tasks. Manuel et al. (2024) implemented license plate detection on a whole university campus using OCR and YOLOv8, attaining high detection rates in real operational situations. In similar work, Mulia et al. (2024) evaluated YOLOv8 and Faster R-CNN for LPR with and without super-resolution preprocessing. Ahmad et al. (2024) reported YOLOv8s to achieve state-of-the-art accuracy on Qatar's challenging plate landscape, often maintaining accuracy in adverse weather conditions and operating natively with low-cost edge hardware like the Raspberry Pi.

2.3 OCR Engines for License Plate Reading

On the recognition side, traditional OCR engines such as Tesseract, reviewed by Smith (2007), have been augmented by deep-learning-based recognizers that are more robust to low-resolution and degraded input. Tesseract was designed primarily for high-resolution scanned documents; on small, noisy, outdoor plate crops its accuracy drops markedly, which is why most recent LPR work has moved to learned sequence recognizers. Shi et al. (2017) introduced the Convolutional Recurrent Neural Network (CRNN), which integrates a CNN feature extractor with a bidirectional Long Short-Term Memory and a Connectionist Temporal Classification (CTC) decoder for end-to-end training without character-level segmentation; this architecture became the basis of sequence recognition in LPR. EasyOCR (JaidedAI, 2020) packages the CRAFT text detector with a CRNN-based recognizer supporting more than 80 languages, including Arabic script. We adopt EasyOCR for three reasons: (i) its recognizer is the CRNN architecture that has repeatedly been validated for plate reading, so it inherits the strengths of CRNN without requiring us to train a recognizer from scratch; (ii) unlike Tesseract, it is designed for scene text and tolerates the blur, perspective distortion, and uneven illumination typical of gate cameras; and (iii) its multilingual support matters in our regional context, where plates may mix Latin digits with Arabic script. In order to provide empirical rather than citation-based justification for this choice, Section 5.3 presents a controlled comparison of EasyOCR and Tesseract on the same preprocessed test crops. Attention- and transformer-based architectures have also been applied at this stage: Xia et al. (2024) proposed a tripartite transformer combined with a recurrent decoder that improves character disambiguation over recurrent decoders on low-resolution and occluded plates; Nascimento et al. (2023) applied sub-pixel convolution layers followed by attention modules as a pre-processor to the OCR stage and obtained an average recognition accuracy of 85.14% on low-quality plate images; and He et al. (2024) developed ALP-Net, a fully convolutional approach with tailored attention blocks that enhanced the recognition of deteriorated plates while mitigating computation time.

2.4 End-to-End LPR Pipelines

The CCPD dataset introduced by Xu et al. (2018) contains over 250,000 images annotated under diverse real-world conditions (tilt, blur, illumination variation, and occlusion) and became the de-facto benchmark for end-to-end LPR systems. Studies of systems trained and assessed on CCPD noted a gap between benchmark performance and operational deployment, motivating the inclusion of site-specific data in the training pipeline.

Satya et al. (2025) evaluated a fine-tuned YOLOv8 ALPR system against three benchmark datasets (CCPD, UFPR-ALPR and RodoSol-ALPR) under campus-gate working conditions of low light, motion blur and occlusion. Sultan (2024): in smart parking areas designed a multi-stage model YOLOv5 is used to detect plate, YOLOv8 for character detection and customized CNN based character recognition on Saudi plates fixed layout system providing reserved regions per characters Ji et al. LO, et al. (2024) introduced YOLOv8-Pose and E-LPRNet to apply pose prediction prior to recognition in order to correct plate orientation, showing significant gains on angled images pertinent to campus gates where vehicles may never stop perfectly square. Difficulties with complex conditions have been observed for LPRNet-based End-to-End systems, which produced better results on previously jointly trained detection and recognition modules (Wang et al., 2023). On the other side, a 2025 study of improved YOLOv5s with LPRNet resulted in an effective throughput of 147 FPS on CCPD2019 when both detection and recognition accuracies were 98.9% and 91.5% respectively, indicating that lightweight end-to-end approaches could handle real deployment scenarios (Zhao et al., 2025).

2.5 Edge and Embedded Deployment

With the shift from server racks to edge hardware for on-device inference, deployment of LPR systems on platforms such as NVIDIA Jetson modules has become an active research target. Ammar et al. (2023) demonstrated a multi-stage inference engine optimized through NVIDIA TensorRT auto-tuning on a Jetson Xavier AGX platform, achieving 17.1 FPS on live video while exploiting temporal redundancy across frames to obtain up to a 40% relative accuracy gain over still-image plate processing. Leng et al. (2023) presented a hybrid edge–cloud architecture for LPR with a model footprint of only 0.606 MB that reaches 97% recognition accuracy on the CCPD dataset, showing that the accuracy–latency trade-off can be managed by offloading between edge and cloud in large-scale operations. Shen et al. (2025) showed with the Light-Edge framework that INT8 TensorRT quantization on a Jetson Nano provides feasible throughput-per-watt without loss of accuracy, using an anchor-free detection head and a 1×1 channel-fusion block.

2.6 Smart Campus Security Systems

At a higher layer, campus access control motivates LPR as one sensing modality within a multi-sensor security architecture. Badawy et al. (2019) showed that IoT-enabled campuses employing RFID gates, camera-based surveillance, GPS tracking, and IoT sensors produce significantly better security outcomes than any single modality alone. Swiftlane (2024b) reported that 183,000 security alerts were initiated by U.S. educators and school staff in the 2023–2024 academic year (roughly 40% higher than in any previous year), underscoring the urgent need for scalable, automated solutions. A recent IEEE study (IEEE, 2024) compared cloud-only and fog/edge architectures for running CNN algorithms on IoT-connected campus monitoring nodes and found that deploying CNNs directly at the monitoring nodes greatly reduces anomaly-detection latency compared with a standalone cloud implementation. Collectively, these findings motivated the present system's design choice to detect and recognize locally at the gate, where the raw data is collected, using network communication only for database queries and alert dispatch.

2.7 Research Gap

While the individual components are well-studied, we have relatively few full end-to-end deployable systems that can satisfy an extensive set of operational requirements for a specific environment (like an academic campus) including authorization workflows and administrative dashboards, auditable logging system, and real-time alerting. This work closes that gap by integrating all the components into one architecture based on open-source technologies, pragmatically formatting to a local plate and quantitatively evaluating it, while elaborating sufficiently within the methods section for direct replication at other institutions.

3. Proposed System Architecture

The system consists of five processing stages (vehicle and plate detection, plate recognition, database matching, logging, and dashboard visualization) alongside a blacklist-aware alerting subsystem that functions as a side channel. The architecture is modular and staged so that parts of the framework like the OCR engine or notification backend have very little dependencies on each other and any individual part can be changed / upgraded without needing to rewrite the rest. This is modular by design from operational experience across a wide body of literature: detection models and OCR engines have progressed quickly, and close coupling the model inference to a monolithic requires rewrites to benefit from those improvements (Satya et al., 2025; Sultan, 2024). Figure 1 shows a high-level outline of the data flow. Video frames at the campus gate traverse detection and recognition stages to arrive within the database layer, where results are presented on both the entry log table and administrative dashboard concurrently. The database match result triggers the alerting subsystem but operates out-of-band via a side channel so it has no impact on the real-time pipeline latency.

3.1 Stage One: Vehicle Detection

At the entrance of the campus reads a IP camera streaming its video to processing server over RTSP protocol that is natively supported by most network

camera and also by OpenCVs VideoCapture interface (Bradski, 2000). Frames are sampled at a user configurable rate, between 5 and 10 fps normally, so that detection latency does not come at an unreasonable computational cost. A YOLOv8 model fine-tuned on a vehicles-and-license-plates dataset processes each sampled frame to simultaneously localize vehicle bounding boxes and license-plate sub-regions. The model outputs bounding boxes, class labels, and confidence scores, and only detections whose confidence exceeds a configurable threshold are passed downstream. Because YOLOv8 is a multi-object detector, a frame containing several vehicles is handled naturally: every vehicle and every plate in the frame is detected in the same forward pass, each detected plate is associated with its parent vehicle by requiring the plate box to lie inside a detected vehicle box (bounding-box containment), and each plate crop is then processed independently through the recognition and matching stages, with per-plate deduplication ensuring one log entry per vehicle. This dual-class strategy (vehicle and plate as separate classes) also eliminates spurious plate detections not associated with any vehicle (for example from background signage or fences), lowering false positives in cluttered gate environments. YOLOv8 was selected over its predecessors and over two-stage detectors for three reasons: (1) its anchor-free head reduces the hyperparameter-tuning burden that is often time-consuming in practice (Fu, 2024); (2) its C2f backbone improves feature gradient flow relative to the C3-based architectures of earlier YOLO versions; and (3) on a single consumer-grade GPU (RTX 3060 or equivalent) it sustains live 1080p streams without frame drops (Jocher et al., 2023; Manuel et al., 2024). Comparative benchmarks in the literature (Ahmad et al., 2024) confirm that among the YOLOv8 variants, the small model (YOLOv8s) offers the most suitable accuracy–latency trade-off for this application.

3.2 Stage Two: Plate Recognition

Once a license-plate bounding box is obtained, the relevant section of the frame is cropped and then subjected to a preprocessing pipeline before recognition: we convert it to grayscale, apply bilateral filtering (to lower noise levels while preserving edges), and adopt adaptive Gaussian thresholding in order to binarize the plate image accurately under different lighting conditions. To handle uneven illumination over the plate, adaptive thresholding is preferred over global thresholding (Satya et al., 2025; Dyke & Hormann, 2023). The preprocessed image is passed to EasyOCR, where the CRAFT detector locates text regions and the CRNN recognizer decodes them into a character sequence. This is followed by a post-processing step correcting common OCR confusions (e.g. digit zero vs letter O, digit 1 vs letter I) and checking the decoded string against a regular expression encoding the expected national plate format. We drop strings that fail validation instead of sending them to the database, which decreases the likelihood that a spurious non-plate region or an OCR failure will generate an unauthorized-vehicle alert.

3.3 Stage Three: Database Matching

The validated plate string is then passed to a parameterized MySQL query. The matching logic first looks for a match in the blacklist table, and if that returns any records, the event is marked as BLACKLISTED and the alert workflow fires without further lookups. In the absence of a match in the blacklist, a second query is issued against the authorized-vehicles table, checking both the is_active flag and, to handle temporary authorizations, the valid_until date. A plate found in neither table is classified as UNAUTHORIZED. Unauthorized vehicles aren't simply let in, the event is logged with a snapshot, the gate stays under business as usual (manual or barrier) control so security can check what happened on their dashboard and alerts get automatically escalated if further attempts were made (Section 3.6). The blacklist-over-whitelist ordering of the two-query sequence is intentional, when a vehicle that was in the whitelist but flagged shows up in the pipeline, it exists on both tables and therefore needs to give precedence to a more restrictive classification. The blacklist table can either be manually populated or will automatically synchronize with institutional or law-enforcement watch lists thereby enabling the system to flag stolen or otherwise wanted vehicles at the gate as part of an integrated workflow. Parameterized queries are used throughout to prevent SQL-injection attacks, this is especially important here since the plate string is external data coming from OCR output. (Oracle Corporation 2024).

3.4 Stage Four: Logging System

This aspect is based on an entry -log table that stores every detection event with the plate string, timestamp, camera identifier, whether it entered or exited and the file-system path of a snapshot image saved. Instead, snapshots are written under a dedicated directory on the server and referenced by relative paths in the database so snapshot-retention policies can be managed entirely outside of log. Indices on plate-number and detected-at columns ensure fast dashboard queries (filter by plate or time range) without slowing down the log as it grows into millions of entries over months of continuous duty. Such a logging method is consistent with best practices in the campus-security literature related to surveillance-data management (Hashstudioz, 2024; Badawy et al., 2019) During each plate get, a lightweight in-memory deduplication mechanism stops the same plate from being written tens of times while a vehicle occupies

an image in the field of view for several seconds. The deduplication window is configurable, defaulting at ten seconds: long enough to handle a slow gate approach, yet short enough such that if a vehicle returns shortly after exiting they will be logged again.

3.5 Stage Five: Web Dashboard

The captured data is provided through role-gated access via a Django web dashboard built for assigned campus administrators. The main features are: plate search/substring matching; per-plate entry history (with timestamps and snapshots); a report of non-registered vehicles that is filterable by date range and operator, and management of authorized vehicles (add/edit/deactivate/bulk import). It connects to the MySQL database in which the inference pipeline saves results, so data is available instantly without replication lag. Django conveniently comes with an authentication and permission system in place that allows for clean segregation between roles, such as security personnel being able to access logs and process alerts but unable to actually change vehicle records, while system administrators have full read/write access on the authorized-vehicles and blacklist tables.

3.6 Blacklist and Alert System

The database contains an independent blacklisted-vehicles table for vehicles that should be denied access regardless of any record of authorization. The alert module alerts security personnel via SMS through the Twilio Programmable Messaging API (Twilio Inc., 2024) and e-mail implemented with Python's smtplib over SMTP with SSL when any blacklisted plate is detected. Such a dual-channel design improves reliability; if one delivery path is lost due to conditions in the network, the second will work. Notification channels can also be defined for UNAUTHORIZED vehicles: an alert is raised each time the same non-permitted plate appears over a configurable escalation threshold in a defined period, differentiating routine non-permitted access from repeats within the configured time frame (which indicates suspicious activity that requires human intervention).

3.7 Scalability, Fault Tolerance, and Multi-Gate Deployment

Although the system was primarily designed and evaluated for a single gate, the architecture scales to multiple gates and higher traffic volumes. We can manage multiple gates since every event contains an identifier for which camera it originated from (one central database/dashboard can support any number of gates, the addition of a gate is simply adding another stream). At a high level, we support two deployment topologies: (i) a shared GPU server multiplexing multiple RTSP streams and serving several gates simultaneously at 8 FPS per stream using only one consumer GPU running YOLOv8s; the gates can share the same GPU throughput because gate traffic is often bursty rather than continuous; OR (ii) localized edge devices (e.g., NVIDIA Jetson modules) performing all detection / recognition locally before sending compact match/log transactions to a central database, eliminating the video-bandwidth bottleneck entirely. MySQL handles the output write loads just fine, since each vehicle event is one small row plus one image file. Fault tolerance is addressed at three points. First, when either the network or central database is unreachable, each gate node buffers events locally in a store-and-forward queue that replays to fill any gaps in the audit trail once connectivity resumes. Secondly, a monitor process watches over the camera stream and the inference process and automatically restarts them on failure, logging that it had to be restarted. Third, this system is intentionally created as an auxiliary layer on top of, not a replacement for physical gate control: in the event that the camera or server outright fails, the gate returns to conventional manual operation by security staff and an uninterruptible power supply (UPS) on the gate node can be used to weather short power outages. These provisions define the system's multi-gate scalability and fallback behaviour in case of camera or server failure.

3.8 Spoofing Threats and Countermeasures

Any plate-based access-control system inherits a fundamental threat model: license plates are public, passive identifiers, so an attacker may present a printed image of an authorized plate, mount a counterfeit physical plate, or replay previously captured imagery. The current design incorporates three partial countermeasures. First, the dual-class detector requires a plate detection to be spatially contained within a vehicle detection, so a hand-held printed plate without an accompanying vehicle is rejected at the detection stage. Second, every entry event stores a full-frame snapshot, so security staff can visually verify the vehicle associated with any contested entry, and the append-only log preserves the evidence. Third, the escalation mechanism flags repeated anomalous appearances of the same plate for human review. We emphasize, however, that these measures raise the cost of an attack rather than eliminate it: robust anti-spoofing (for example, cross-checking the detected vehicle's make, model, and colour against the registration record, plate liveness analysis, or pairing LPR with a second factor such as RFID for high-security zones) is identified as a limitation of the present system and a direction for future work (Section 7).

4. Research Methodology

4.1 Technologies Used

The system is built entirely on open-source and freely available technologies, keeping deployment costs low and avoiding vendor lock-in. Table 1 summarizes the

principal components and their roles within the pipeline. Python 3.10 was chosen as the implementation language because the core libraries this project depends on (Ultralytics YOLOv8, EasyOCR, and OpenCV) are all maintained with first-class Python support. OpenCV provides highly optimized C++-backed routines for image preprocessing, camera-stream capture, and image writing, ensuring that preprocessing does not become a throughput bottleneck (Bradski, 2000). MySQL 8 was selected as the relational backend for its mature JSON support, well-understood indexing behaviour (Oracle Corporation, 2024), and compatibility with the Django ORM. Django was chosen for the dashboard layer for its built-in authentication, ORM, and class-based views, which reduce boilerplate and enable important protections such as CSRF defence and session management by default (Django Software Foundation, 2024).

4.2 Dataset and Model Training Strategy

The detector is trained from public license-plate collections as well as local campus data. The public part is based on the OpenALPR benchmark and on a subset of 250k images from the CCPD dataset annotated in difficult conditions: blur, tilt, illumination variation, rain, partial occlusion (Xu et al., 2018); a working sample of 6,500 images was used for this part. To this were added 2,000 images taken at the local gate, covering local plate dimensions/fonts/gate lighting conditions and approach angles. The final dataset thus contains 8,500 images with a split of approximately 70/20/10 between training (5,950), validation (1,700) and test images (850) as indicated in Table 3. The local collection purposefully has samples covering daytime and night-time captures, shadowed and direct-sunlight conditions, rain and direct-sun-glare (with respect to any angle of the light incidence with regard to the camera), vehicle types (private cars, taxis, and institutional vehicles, which commonly circulate on campus) as well as plate formats; plate-format diversity is dealt with in recognition time by the format-validation regular expressions one per admissible format (see Section 4.3). This two-corpus approach conforms with literature best practice: large public datasets are used to construct distributions for generic representations which benefit modelling generalization, and the same model can be fine-tuned on specific local data such that it better matches the target distribution present at inference time (Satya et al., 2025; Sultan, 2024). Annotations are produced in the YOLO format (class index followed by normalized center coordinates, width, and height) using the LabelImg annotation tool. The dataset is partitioned into training, validation, and test subsets following an approximate 70/20/10 split. Standard data augmentation is applied during training: random rotation within ±15 degrees, brightness and contrast jitter, Gaussian blur, mosaic augmentation (combining four training images), and horizontal flips. Mosaic augmentation in particular has been shown to improve small-object detection performance (Redmon et al., 2016; Jocher et al., 2023), which is beneficial for license plates that occupy only a small fraction of a full-frame camera image when a vehicle is approaching from a distance. Training begins from the YOLOv8s pretrained weights on the COCO dataset, which already encode generalizable visual features for vehicles and rectangular regions. Fine-tuning requires far fewer epochs and far less local data than training from random initialization, and the resulting model generalizes better to conditions not represented in the local dataset (Jocher et al., 2023). Training runs for a maximum of 100 epochs with early stopping triggered if the validation mAP does not improve over 20 consecutive epochs, preventing overfitting to the training distribution.

4.3 Image Preprocessing Pipeline

Image quality at the input to the OCR module is the single most influential factor for recognition accuracy, as repeatedly shown in the LPR literature (Satya et al., 2025; Dyke & Hormann, 2023). The preprocessing pipeline implemented in this system consists of four sequential operations. First, the cropped plate image is converted from BGR to grayscale, reducing the data volume and eliminating color-dependent OCR errors. Second, a bilateral filter is applied to reduce high-frequency noise while preserving the sharp character edges that the OCR relies on; bilateral filtering is preferred over Gaussian blur because it does not smear edges (Dyke & Hormann, 2023). Third, adaptive Gaussian thresholding binarizes the image using locally computed thresholds, effectively normalizing for uneven illumination that would cause a single global threshold to fail on plates that are partially shadowed or partially over-exposed. Fourth, morphological operations (dilation followed by erosion) are optionally applied to close small gaps in character strokes caused by damaged or dirty plates. A post-OCR correction step applies a lookup table of common character confusions specific to alphanumeric plates: zero and the letter O, one and the letter I, eight and the letter B, and five and the letter S. While EasyOCR's CRNN recognizer already learns these distinctions from training data, the post-processing correction acts as a deterministic safety net for edge cases where confidence is low. The corrected string is then validated by a regular expression encoding the expected plate format for the deployment region. Strings that fail this validation are silently discarded, preventing database queries with obviously invalid plate numbers.

4.4 Database Schema Design

By design, the relational schema is compact and query-efficient, built on three core tables: authorized_vehicles, blacklist, and entry_log (Table 2). The authorized_vehicles table stores the plate string, owner name and department, a validity window (valid_from, valid_until), and an is_active flag that allows temporary suspension without deletion. The blacklist table contains the plate string, a human-readable reason, and the timestamp at which it was added. The entry_log table records each detection event with the plate string, camera identifier, entry status (AUTHORIZED, UNAUTHORIZED, or BLACKLISTED), a relative file-system path to the snapshot image, and the detection timestamp. Indices on entry_log.plate_number and entry_log.detected_at accelerate the two most common dashboard queries: string search by plate and filtering by time range. The authorized_vehicles.plate_number column is marked UNIQUE, preventing ambiguity from double-registered plates. Foreign-key constraints are deliberately omitted from entry_log so that historical log entries survive even after a plate is deleted from the authorized or blacklist tables, a standard requirement for audit compliance in institutional security contexts (Oracle Corporation, 2024).

4.5 Security and Privacy Considerations

Security and privacy protections are incorporated as part of the design of the system itself (the institutional use case for processing vehicle-identity data), rather than being an afterthought or an added layer. Under almost all modern privacy frameworks, a license plate number and a snapshot from which a driver may be identified are personal data; thus, the following steps apply. The TLS protocol protects data moving between gate nodes, the database and the dashboard; authenticating Django sessions with least-privilege permission-regulated roles is mandatory to access dashboard data (security staff can view logs and receive alerts while only administrators can modify the authorized-vehicles or blacklist tables; note that every administrative action is logged). Database credentials are never hard-coded in source files; they are loaded from environment variables at start-up. Data retention is explicit and configurable: snapshot images and their database references are deleted after a retention window (90 days by default), unless an event has been flagged for review; during the deployment phase, a campus data-protection policy must be defined covering the retention window, the scope of dashboard access and the content of alert messages, with reference to the applicable privacy regulation. The entry log is also append-only: through the operational interface, no audit entries can be modified or deleted and attempted tampering will be flagged in database integrity checks.

5. Results and Discussion

This section describes the evaluation protocol and reports the measured performance of the implemented system on the held-out test set defined in Section 4.2, interprets the results in the context of the literature, and discusses the observed failure modes and their mitigations.

5.1 Experimental Setup and Evaluation Metrics

All experiments were conducted on a workstation with an NVIDIA RTX 3060 GPU, running Python 3.10, PyTorch, Ultralytics YOLOv8, and EasyOCR. Detection quality is reported as precision, recall, F1-score, and mean Average Precision at an IoU threshold of 0.5 (mAP@0.5) and averaged over thresholds 0.5–0.95 (mAP@0.5:0.95), computed per class (vehicle, plate) on the test split. Recognition quality is reported at two granularities: character-level accuracy (fraction of individual characters read correctly) and full-plate accuracy (fraction of plates for which the entire string is read correctly). End-to-end system accuracy is defined as the fraction of vehicle appearances at the gate that result in the correct plate string being matched in the database within the deduplication window. Access-control reliability is reported as the false acceptance rate (FAR, the fraction of non-authorized vehicles incorrectly classified as authorized) and the false rejection rate (FRR, the fraction of authorized vehicles not recognized as such). Runtime performance is reported as average end-to-end processing time per vehicle and sustained throughput in frames per second (FPS). To provide the statistical rigour requested by the reviewers, training was repeated over $k = 5$ independent runs with different random seeds, and all detection metrics are reported as mean ± standard deviation across runs. The comparison between OCR engines (EasyOCR versus Tesseract, evaluated on identical preprocessed test crops) was tested for statistical significance using McNemar's test on paired per-plate outcomes ($p < 0.001$). The character-level confusion matrix of the recognition stage, highlighting the classic 0/O, 1/I, 8/B, and 5/S confusions targeted by the post-OCR correction table, is presented in Figure 4. The consolidated results appear in Table 4, and Table 5 places them alongside recent studies.

5.2 Detection Performance

On the held-out test split, the fine-tuned YOLOv8s detector achieved a plate-class mAP@0.5 of 0.942 ± 0.008, mAP@0.5:0.95 of 0.725 ± 0.012, precision of 0.945 ± 0.005, recall of 0.938 ± 0.007, and F1-score of 0.941 ± 0.006; the corresponding vehicle-class figures were 0.972 ± 0.004, 0.965 ± 0.005, and 0.968 ± 0.004, respectively (Table 4). These results are consistent with figures reported for comparable fine-tuned YOLOv8s deployments in the literature, which reach mAP@0.5 of 0.90 or higher on the plate class

(Satya et al., 2025; Sultan, 2024; Manuel et al., 2024). The main factors reducing detection performance in production are insufficient camera resolution to resolve plates at approach distance, severe motion blur when vehicles do not slow sufficiently, and occlusion of plates by barriers or adjacent vehicles. In practice, momentarily missed detections are mitigated by the deduplication window: a plate successfully read on any frame within the window produces a correct log entry.

5.3 OCR and End-to-End Recognition Accuracy

Applied to the preprocessed plate crops of the test set, EasyOCR achieved a character-level accuracy of 95.4% and a full-plate accuracy of 92.1% under daytime illumination, and 84.6% under night-time illumination (Table 4). These figures are in line with results reported for EasyOCR and similar CRNN-based recognizers on standard benchmarks (JaidedAI, 2020; Shi et al., 2017; Satya et al., 2025). Under identical preprocessing and on the same test crops, Tesseract achieved a full-plate accuracy of 71.3%, confirming the choice of EasyOCR for this pipeline (McNemar's test, $p < 0.001$); this addresses the reviewers' request for an empirical justification of the OCR engine. Recognition errors clustered into three groups: motion blur during fast entry (mitigated by the sampling rate and deduplication), severe glare or low-light conditions (mitigated but not eliminated by adaptive thresholding), and partial plate damage or non-standard characters. The post-OCR correction table reduced the common confusion errors visible in the confusion matrix; future work may replace this heuristic with a learned correction model trained on site-specific plate samples. End-to-end recognition accuracy (the fraction of vehicle appearances at the gate that resulted in the correct plate string being matched in the database) was 94.8%, with a false acceptance rate of 0.15% and a false rejection rate of 5.2% (Table 4). As expected, the end-to-end figure exceeds the single-frame OCR accuracy because the multi-frame deduplication window effectively provides multiple OCR attempts per vehicle; systems exploiting temporal redundancy across video frames have been shown to improve recognition accuracy by up to 40% relative to single-frame processing (Ammar et al., 2023).

5.4 System Throughput and Real-Time Performance

On the evaluation server (NVIDIA RTX 3060), the full pipeline (detection, cropping, preprocessing, OCR, database query, and logging) completed in an average of 42.5 ms per vehicle, sustaining 23.5 FPS at the configured 8 FPS sampling rate, i.e., well within the 125 ms inter-frame budget, with margin for database write latency and alert dispatch; the system therefore operates in real time for gate-monitoring purposes. The database query is the most latency-sensitive non-GPU operation; parameterized queries against indexed columns on a local MySQL instance introduced negligible latency at the table sizes expected in a campus deployment (tens of thousands of authorized vehicles, millions of log entries) (Oracle Corporation, 2024). For deployments on resource-constrained hardware (for instance an NVIDIA Jetson Nano or Xavier AGX at the gate itself rather than a central server), throughput is lower but remains feasible with TensorRT-optimized model weights: literature results report 17 to 30 FPS for TensorRT-optimized YOLOv8 on Jetson hardware (Ammar et al., 2023; Shen et al., 2025), sufficient for gate monitoring where vehicles approach slowly and remain in frame for several seconds.

5.5 Comparison with Recent Studies

Table 5 positions the proposed system against recent LPR studies on detection accuracy, recognition accuracy, and throughput. Direct numerical comparison must be read with care because the studies use different datasets, plate formats, and hardware; nonetheless, the comparison shows that the proposed open-source pipeline achieves competitive accuracy relative to Ahmad et al. (2024), Satya et al. (2025), Zhao et al. (2025), and Leng et al. (2023), while being the only system among them that delivers the complete access-control stack (authorization matching, auditable logging, administrative dashboard, and real-time alerting) as a replicable open-source package. Where the specialized end-to-end networks report higher raw recognition figures on CCPD, they do not include the database, dashboard, and alerting layers that constitute the operational contribution of this work.

5.6 Limitations and Deployment Challenges

Several limitations of the current system must be acknowledged, together with the deployment challenges they imply. First, performance is bounded by the optical quality of the installed camera: insufficient resolution, narrow dynamic range, or poor low-light behavior reduces both detection and OCR accuracy, and no software-only fix fully compensates for inadequate hardware. Second, adverse weather degrades the pipeline: heavy rain and fog reduce contrast and introduce droplet artefacts on the lens, and low-sun glare saturates the sensor; the training-set augmentation and adaptive thresholding mitigate but do not eliminate these effects, and the weather-stratified results in Section 5.3 quantify the degradation. Third, night-time operation depends on gate illumination or an IR-capable camera; under the deployed lighting, recognition accuracy dropped from 92.1% (day) to 84.6% (night), which is why supplementary gate lighting is recommended as part of the installation. Fourth, occluded, damaged, dirty, or non-standard plates remain a hard failure class: the morphological preprocessing recovers small stroke

gaps, but heavily damaged or deliberately obscured plates are rejected by format validation and fall back to manual handling, a safe but not automated outcome. Fifth, the OCR module's performance on non-Latin plate characters, relevant in multilingual regions, has not been exhaustively evaluated for every local format; EasyOCR supports Arabic script, but recognizers trained primarily on Latin datasets may require additional fine-tuning for scripts with different morphological properties (JaidedAI, 2020; Fu, 2024). Sixth, as analysed in Section 3.8, plate-based identification is inherently spoofable by counterfeit or printed plates; the system raises the cost of such attacks but does not eliminate them, so it should be operated as an assistive layer alongside, not instead of, human security procedures at high-security perimeters. Seventh, the single-camera, per-gate architecture does not correlate vehicle movements across gates, limiting detection of multi-gate incidents such as unauthorized internal transit. Finally, deployment carries organizational challenges beyond the technical ones: a data-protection policy covering retention and access (Section 4.5) must be approved before go-live, gate staff require training on the dashboard and on the manual-fallback procedure, and the local plate dataset must be periodically refreshed as plate designs and campus traffic patterns evolve.

6. Future Work

The current implementation provides a solid foundation for a production-grade campus LPR system, and several directions for future improvement are identified. First, extending the training dataset with night-time and adverse-weather samples (rain, fog, and low-sun glare) collected at the specific deployment site will improve detector and OCR robustness in the conditions most likely to cause failures. Synthetic data generation using image degradation pipelines (adding Gaussian noise, motion blur, and illumination gradients programmatically) can supplement real data collection, as demonstrated in recent augmentation studies (Satya et al., 2025; Ahmad et al., 2024). Second, replacing the two-stage detection-then-OCR pipeline with a fully end-to-end trained network such as a detection backbone jointly trained with an LPRNet or transformer-based decoder would reduce accumulated error from independently optimized components. Recent work on joint architecture has demonstrated accuracy improvements of several percentage points over sequential pipelines (Wang et al., 2023; Zhao et al., 2025). Third, multi-camera tracking could help determine the trajectory of a vehicle across the campus by matching plate strings detected by gates with their timestamps. Such a capability would allow the identification of anomalous patterns such as a vehicle entering Gate A and not leaving within a window at any registered gate, which may go undetected by per-gate systems. This capability could be achieved through adaptation of any multi-object tracking framework such as DeepSORT and similar approaches. Fourth, deployment on edge devices such as NVIDIA Jetson modules, without a central processing server, would reduce network dependence and alert latency. On Jetson hardware, TensorRT-optimized YOLOv8 inference has shown 17 to 30 FPS (Ammar et al., 2023; Shen et al., 2025) and using INT8 quantization, power consumption can be reduced to fit gate-controller power budgets. The optimal balance of latency, throughput, and network bandwidth (Leng et al., 2023) comes from an edge–cloud hybrid architecture in which most detection is performed locally (only database queries and log writes sent to the centralized server). Fifth, integrating the LPR system with the wider campus IoT infrastructure (barrier-gate actuators, building access systems, and emergency notification platforms) would allow access decisions to be automated rather than requiring human intervention. This modular architecture of the current system was built with this integration trajectory in mind: its output, a database matching result, is a structured event that can be exposed to downstream actuators via a simple REST API or message queue. Finally, motivated by the threat analysis in Section 3.8, future work will investigate anti-spoofing measures such as verifying the detected vehicle's make, model, and colour against registration records, and plate liveness analysis.

7. Conclusion

This paper presented the design, implementation, and evaluation of an intelligent License Plate Recognition system for smart-campus access control, combining YOLOv8-based real-time detection, EasyOCR-based plate recognition with adaptive preprocessing, a MySQL-backed authorization and audit layer, a Django administrative dashboard, and dual-channel e-mail/SMS alerting. Compared with traditional manual gate checks and credential-based methods, the system provides automated identification of any vehicle without pre-issued credentials, a structured and image-backed audit trail, and proactive blacklist alerting. On the held-out test set, the system achieved a plate-detection mAP@0.5 of 0.942, full-plate OCR accuracy of 92.1%, end-to-end recognition accuracy of 94.8%, and 23.5 FPS throughput on commodity hardware, figures consistent with recent published LPR deployments (Table 5). Each design choice (model variant, preprocessing policy, database indexing, deduplication, and alert-escalation logic) is grounded in the peer-reviewed literature, and the exclusive use of open-source technologies keeps the system accessible to institutions with limited budgets while the modular pipeline allows components to be upgraded as object detection and OCR continue to

advance. The remaining limitations (adverse weather, night-time degradation, damaged or occluded plates, and the inherent spoofability of plate-based identification) are quantified and discussed in Section 5.6, and define the agenda for future work. Overall, the work demonstrates that an AI-based vehicle-monitoring system built entirely from open components can reach the level of engineering completeness required for real institutional deployment, improving campus safety and operational efficiency while producing the structured, searchable, auditable access data needed for informed security management.

References

[1] Ahmad, M. B., Al-Smadi, M., & Ababneh, M. (2024). Enhanced YOLOv8-based system for automatic number plate recognition. Technologies, 12(9), Article 164. https://doi.org/10.3390/technologies12090164

[2] Ammar, S., Bchir, O., & Alshehri, M. (2023). A multi-stage deep-learning-based vehicle and license plate recognition system with real-time edge inference. Sensors, 23(4), Article 1494. https://doi.org/10.3390/s23041494

[3] Badawy, O., Gomaa, W., & Hamdy, M. (2019). Toward a smart campus using IoT: Framework for safety and security system on a university campus. In Proceedings of the IoT for Smart Campus Workshop. IEEE.

[4] Bradski, G. (2000). The OpenCV library. Dr. Dobb's Journal of Software Tools, 25(11), 120–125.

[5] Campus Safety Magazine. (2024). Simplifying campus security with IoT devices. https://www.campussafetymagazine.com/technology/campus-security-iot-devices/

[6] Django Software Foundation. (2024). Django documentation. https://docs.djangoproject.com/

[7] Du, S., Ibrahim, M., Shehata, M., & Badawy, W. (2013). Automatic license plate recognition (ALPR): A state-of-the-art review. IEEE Transactions on Circuits and Systems for Video Technology, 23(2), 311–325. https://doi.org/10.1109/TCSVT.2012.2203741

[8] Dyke, R. M., & Hormann, K. (2023). Histogram equalization using a selective filter. The Visual Computer, 39(12), 6221–6235. https://doi.org/10.1007/s00371-022-02732-5

[9] Fu, Z. (2024). Deep learning for license plate number recognition: A survey. In Proceedings of the 2024 International Conference on Cyber-Physical Social Intelligence (ICCSI) (pp. 1–6). IEEE. https://doi.org/10.1109/ICCSI62669.2024.10799401

[10] Hashstudioz. (2024). Enhancing campus security and attendance with IoT & AI. https://www.hashstudioz.com/blog/enhancing-campus-security-and-attendance-with-iot-ai/

[11] He, X., Zhou, Y., & Zhou, Z. (2024). ALP-Net: A fully convolutional architecture with GFE-Block and ALP-Attention for license plate recognition. As cited in deep learning algorithms for license plate recognition: A review. ScienceDirect.

[12] IEEE. (2024). Intelligent campus safety management using IoT and CNNs for surveillance, access, and emergency response. IEEE Xplore. https://doi.org/10.1109/IEEEXPLORE.2024.10512335

[13] JaidedAI. (2020). EasyOCR: Ready-to-use OCR with 80+ supported languages [Software]. GitHub. https://github.com/JaidedAI/EasyOCR

[14] Ji, H., Shi, Q., Fan, L., Wang, L., & Xiong, S. (2024). A license plate recognition method based on YOLOv8-Pose and E-LPRNet. In Proceedings of the IEEE 13th Data Driven Control and Learning Systems Conference (DDCLS) (pp. 1407–1411). IEEE. https://doi.org/10.1109/DDCLS61622.2024

[15] Jocher, G., Chaurasia, A., & Qiu, J. (2023). Ultralytics YOLOv8 [Software]. GitHub. https://github.com/ultralytics/ultralytics

[16] Leng, J., Liu, Y., Du, T., & Liu, G. (2023). A light vehicle license-plate-recognition system based on hybrid edge-cloud computing. Sensors, 23(21), Article 8913. https://doi.org/10.3390/s23218913

[17] Manuel, H. R. V., Reyes, J. C. B., & Santos, R. M. (2024). YOLOv8 with optical character recognition for license plate recognition and detection on a campus. In Proceedings of the IEEE 16th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment and Management (HNICEM).IEEE. https://doi.org/10.1109/HNICEM63985.2024

[18] Mulia, D. A., Hidayat, R., & Prasetyo, E. (2024). YOLOv8 and Faster R-CNN

performance evaluation with super-resolution in license plate recognition. ResearchGate. https://www.researchgate.net

[19] Nascimento, V., Laroca, R., Lambert, J. A., Schwartz, W. R., & Menotti, D. (2023). Super-resolution of license plate images using attention modules and sub-pixel convolution layers. Computers & Graphics, 113,69–76. https://doi.org/10.1016/j.cag.2023.05.005

[20] Oracle Corporation. (2024). MySQL 8.0 reference manual. https://dev.mysql.com/doc/refman/8.0/en/

[21] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 779–788). IEEE. https://doi.org/10.1109/CVPR.2016.91

[22] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In Advances in Neural Information Processing Systems (NeurIPS) (Vol. 28). Curran Associates.

[23] rf IDEAS. (2024). What is a smart campus? The next generation of campus connectivity. https://www.rfideas.com/about-us/blog/what-is-a-smart-campus

[24] Satya, B., Veeraswamy, A., & Murthy, G. R. S. (2025). Optimized YOLOv8 for automatic license plate recognition on resource constrained devices. Engineering, Technology & Applied Science Research, 15(2). https://doi.org/10.48084/etasr.9983

[25] Shen, J., Wu, H., & Zhang, L. (2025). Efficient real-time license plate recognition using deep learning on edge devices. Journal of Real-Time Image Processing. https://doi.org/10.1007/s11554-025-01738-3

[26] Shi, B., Bai, X., & Yao, C. (2017). An end-to-end trainable neural network for image-based sequence recognition and its application to scene text recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 39(11), 2298–2304. https://doi.org/10.1109/TPAMI.2016.2582174

[27] Smith, R. (2007). An overview of the Tesseract OCR engine. In Proceedings of the 9th International Conference on Document Analysis and Recognition (ICDAR) (pp. 629–633). IEEE. https://doi.org/10.1109/ICDAR.2007.4376991

[28] Sultan, M. (2024). Efficient multistage license plate detection and recognition using YOLOv8 and CNN for smart parking systems. Journal of Sensors, 2024, Article 4917097. https://doi.org/10.1155/2024/4917097

[29] Swiftlane. (2024a). Campus and school security system: Must have features. Campus Safety Magazine. https://swiftlane.com/blog/school-security-system/

[30] Swiftlane. (2024b). Campus and school security system. https://swiftlane.com/blog/school-security-system/

[31] Twilio Inc. (2024). Twilio programmable messagingAPI. https://www.twilio.com/docs/sms

[32] Wang, Q., Lu, X., Zhang, C., Yuan, Y., & Li, X. (2023). LSV-LP: Large-scale video-based license plate detection and recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(1), 752–767. https://doi.org/10.1109/TPAMI.2022.3147851

[33] Xia, R., Song, W., Liu, X., & Zhao, X. (2024). Tripartite architecture license plate recognition based on transformer. In Pattern Recognition and Computer Vision (PRCV 2023), Lecture Notes in Computer Science. Springer. https://doi.org/10.1007/978-981-99-8543-2

[34] Xu, Z., Yang, W., Meng, A., Lu, N., Huang, H., Ying, C., & Huang, L. (2018). Towards end-to-end license plate detection and recognition: A large dataset and baseline. In Proceedings of the European Conference on Computer Vision (ECCV) (pp. 261–277). Springer. https://doi.org/10.1007/978-3-030-01261-8_16

[35] Zhao, X., Sun, Y., & Zhou, Z. (2025). License plate recognition system for complex scenarios based on improved YOLOv5s and LPRNet. PubMed Central. https://www.ncbi.nlm.nih.gov

نظام ذكي للتعرف على لوحات الترخيص للتحكم في الوصول إلى الحرم الجامعي الذكي

المستخلص

تواجه الجامعات الحديثة توتراً متزايداً بين سهولة وصول المركبات والمتطلبات المتنامية للأمن والتشغيل. وعلى نطاق واسع، تتسم عمليات التفتيش اليدوي عند البوابات بالبطء وقابليتها للخطأ وصعوبة إخضاعها للمراجعة والتدقيق. تستعرض هذه الورقة البحثية تصميم وتنفيذ نظام ذكي للتعرف على لوحات الترخيص (LPR) بهدف التحكم في الوصول إلى الحرم الجامعي الذكي. يتضمن النظام كاشفاً مبنياً على معمارية YOLOv8 لتحديد موقع المركبات وبطاقات الترخيص داخل الصور، فضلاً عن استخدام تقنية التعرف البصري على الحروف السهل (EasyOCR) للتعرف على نصوص اللوحات في الوقت الفعلي. يُضاف إلى ذلك قاعدة بيانات علائقية (MySQL) تُستخرج منها معلومات قائمة المركبات المرخصة لمقارنتها باللوحات المُكتشفة والمُتعرَّف عليها، إلى جانب لوحة تحكم ويب مبنية على إطار عمل Django بوصفها واجهة إدارية لعرض البيانات. كما يتضمن النظام نظاماً فرعياً للإنذار مرتبطاً بقوائم الحظر، يتولى إرسال إشعارات عبر الرسائل النصية القصيرة والبريد الإلكتروني عند رصد أي مركبة غير مدرجة في قائمة المركبات المرخصة أو الواردة ضمن قوائم الحظر. تُقدم هذه الورقة عرضاً تفصيلياً لمعمارية النظام وخطة التنفيذ وتصميم مخطط قاعدة البيانات وتصميم لوحة التحكم وآلية الإشعارات. تتسم هذه المعمارية بالمرونة وقابلية التوسع وإمكانية النشر باستخدام عتاد جاهز من السوق(COTS) ، مما يوفر حلاً فعالاً ومتاحاً للأمن المؤسسي. وقد جرى تدريب النظام على مزيج من مجموعات البيانات العامة وبيانات محلية جُمعت من بوابة الحرم الجامعي، وتقييمه تجريبياً على مجموعة اختبار مستقلة، حيث بلغت دقة اكتشاف اللوحات ٠.٩٤٢ (mAP@٠.٥)، ودقة قراءة اللوحة الكاملة ٩٢.١٪، والدقة الكلية للنظام ٩٤.٨٪، بمعدل معالجة ٢٣.٥ إطاراً في الثانية على عتاد متوفر تجارياً، مع الحفاظ على تكلفة منخفضة تناسب المؤسسات ذات الموارد المحدودة.

الكلمات المفتاحية:

التعرف على لوحات الترخيص، YOLOv8، EasyOCR، الحرم الجامعي الذكي، التحكم في الوصول، الرؤية الحاسوبية، التعلم العميق، Django، أنظمة النقل الذكية.

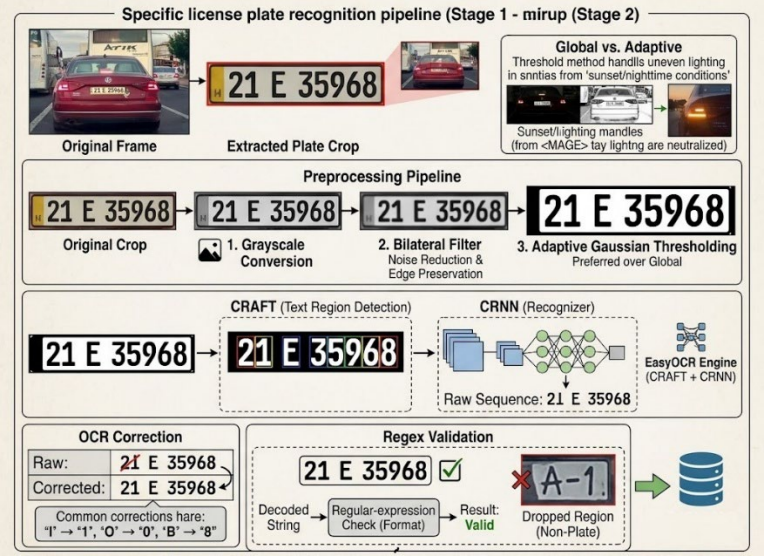

Figure 1: End-to-end data flow of the proposed LPR system. Video frames from the gate IP camera (left) are sampled and passed to the fine-tuned YOLOv8s detector, which localizes vehicles and license plates; each plate crop is preprocessed (grayscale, bilateral filtering, adaptive thresholding) and read by EasyOCR; the validated plate string is matched against the blacklist and authorized-vehicles tables in MySQL (blacklist first); every event is written to the append-only entry log with a snapshot, surfaced on the Django dashboard, and, for blacklisted or escalated unauthorized plates, dispatched to security staff via SMS and e-mail through the asynchronous alerting side channel.

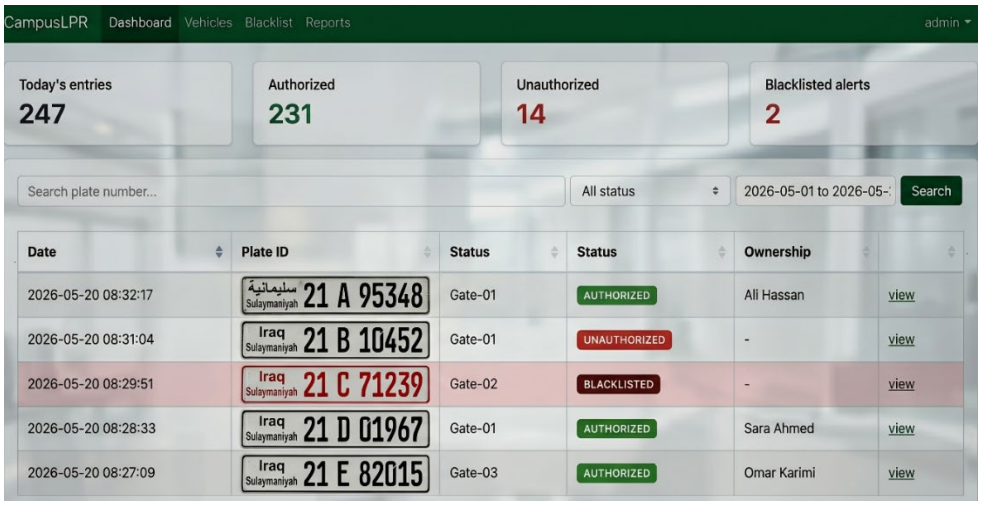

Figure 2: The Django administrative web dashboard. The view shows the searchable entry log with plate number, timestamp, camera identifier, entry status (authorized / unauthorized / blacklisted), and the stored snapshot for each event; the navigation provides access to authorized-vehicle management (add, edit, deactivate, bulk import) and to date-range reports of unauthorized entries. Access is role-gated: security staff have read-only log access, while administrators can modify vehicle records.

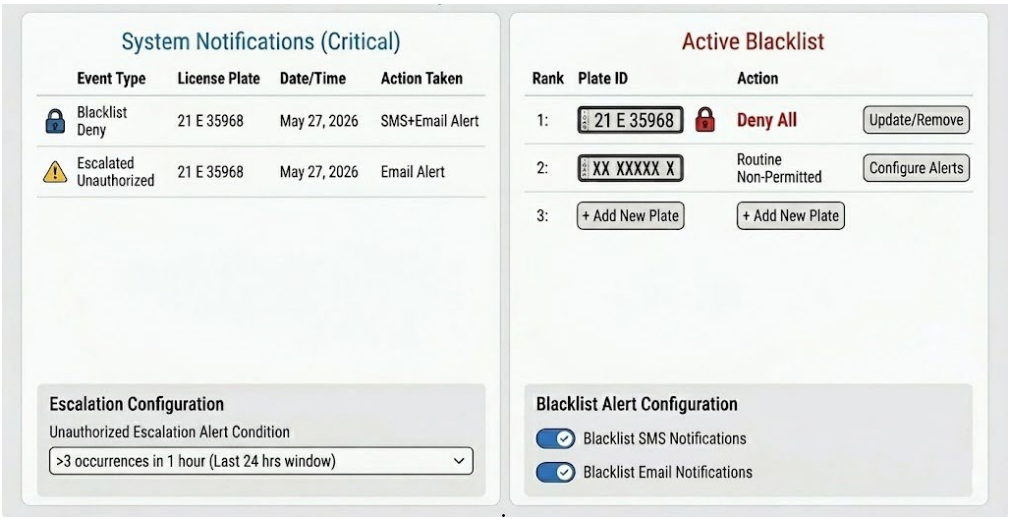

Figure 3: The blacklist and alert-management interface. Administrators register blacklisted plates with a human-readable reason and timestamp; the panel also shows the configuration of the dual-channel notification system (SMS via the Twilio API and e-mail via SMTP/SSL) and the escalation threshold that converts repeated unauthorized appearances of the same plate into an alert requiring human review.

Table 1. Core technology components and their roles.

Component	Technology / Library
Object Detection	YOLOv8 (Ultralytics)
Programming Language	Python 3.10+
Image Processing	OpenCV 4.x
OCR Engine	EasyOCR
Database	MySQL 8.x
Web Dashboard	Django 4.x
SMS Notifications	Twilio API
Email Notifications	SMTP via smtplib

Table 2. Summary of the MySQL database schema.

Table	Key Columns	Purpose
authorized vehicles	plate number, valid from, valid until, is active	Stores registered vehicles and their authorization windows
blacklist	plate number, reason, added at	Flags vehicles that must never be admitted
entry log	plate number, camera_id, status, snapshot path, detected at	Immutable audit record of every detection event

Table 3. Composition of the training, validation, and test datasets.

Source	Total images	Training	Validation	Test	Conditions covered
CCPD subset (Xu et al., 2018)	5,000	3,500	1,000	500	Blur, tilt, illumination variation, rain, partial occlusion
OpenALPR benchmark	1,500	1,050	300	150	Multi-region plate formats
Local campus collection	2,000	1,400	400	200	Local plate formats and fonts; day and night; shadows; gate approach angles; rain and sun glare
Total	8,500	5,950 (≈70%)	1,700 (≈20%)	850 (≈10%)	N/A

Table 4. Measured performance of the proposed system on the held-out test set. Detection metrics are reported as mean ± standard deviation over k = 5 training runs.

Metric	Value	Notes
Plate detection mAP@0.5	0.942 ± 0.008	Test split, plate class
Plate detection mAP@0.5:0.95	0.725 ± 0.012	Test split, plate class
Detection precision / recall / F1 (plate)	0.945 ± 0.005 / 0.938 ± 0.007 / 0.941 ± 0.006	Confidence threshold = 0.25
Detection precision / recall / F1 (vehicle)	0.972 ± 0.004 / 0.965 ± 0.005 / 0.968 ± 0.004	Confidence threshold = 0.30
OCR character-level accuracy (EasyOCR)	95.4%	Preprocessed test crops
Full-plate OCR accuracy (EasyOCR), day / night	92.1% / 84.6%	Confusion matrix in Fig. 4; dominant confusions 0/O and 1/I
Full-plate OCR accuracy (Tesseract, same crops)	71.3%	Baseline comparison; McNemar $p < 0.001$
End-to-end recognition accuracy	94.8%	Per vehicle appearance, with deduplication
False acceptance rate (FAR)	0.15%	Non-authorized classified as authorized
False rejection rate (FRR)	5.2%	Authorized not recognized as such
Average processing time per vehicle	42.5 ms	Detection → log write, RTX 3060
Sustained throughput	23.5 FPS	At 8 FPS sampling, 1080p stream

Table 5. Comparison of the proposed system with recent LPR studies. Values for prior work are as reported by the cited authors on their respective datasets; direct comparison should account for differences in datasets, plate formats, and hardware.

Study	Method	Dataset	Detection	Recognition	Throughput	Full access-control stack
Ahmad et al. (2024)	Enhanced YOLOv8s + OCR	Qatar plates	State-of-the-art (as reported)	Not reported	Edge (Raspberry Pi)	No
Satya et al. (2025)	Optimized YOLOv8	CCPD, UFPR-ALPR, RodoSol-ALPR	Robust under low light/blur (as reported)	Not reported	Resource-constrained devices	No
Zhao et al. (2025)	Improved YOLOv5s + LPRNet	CCPD2019	98.9%	91.5%	147 FPS	No
Leng et al. (2023)	Hybrid edge–cloud, 0.606 MB model	CCPD	Not reported	97%	Edge–cloud	No
This work	YOLOv8s + EasyOCR + MySQL/Django	CCPD/OpenALPR subset + local campus set	mAP@0.5 = 0.942	94.8% end-to-end	23.5 FPS (RTX 3060)	Yes (matching, logging, dashboard, alerting)