

An IoT-Based Smart Parking Management System with Automatic Number Plate Recognition and Environmental Hazard Monitoring

Niraj Bhatta

Department of Computer Engineering
Nepal College of Information Technology
Pokhara University,
Lalitpur, Nepal
niraj.231326@ncit.edu.np

Ravi Yadav

Department of Computer Engineering
Nepal College of Information Technology
Pokhara University,
Lalitpur, Nepal
ravi.231331@ncit.edu.np

Soniya Chand

Department of Computer Engineering
Nepal College of Information Technology
Pokhara University,
Lalitpur, Nepal
soniya.231341@ncit.edu.np

Amit K. Shrivastava

Department of Computer Engineering
Nepal College of Information Technology
Pokhara University,
Lalitpur, Nepal
amit.shrivastava@ncit.edu.np

Abstract—Parking is a growing problem in urbanizing Nepali cities, where many parking lots still rely on manual occupancy recording, vehicle logging, and cash transactions. Although past IoT applications have looked into parking systems, Nepali ANPR, reservation management, and automatic parking payments have been investigated separately in earlier research, there are very few cases where all these aspects are combined together along with environmental hazard detection into a single, low-cost physical model. This paper presents a connected IoT-based smart parking system using ESP32/Arduino sensors and actuators, a React.js dashboard backed by Firebase RTDB, and a tiered Nepali ANPR pipeline. Plate localization and character detection are performed using YOLOv8s, a fine-tuned 34-class CNN, and geometrical reconstruction to assemble individually detected characters into a full plate string. The object detection framework had 95.44% mAP@0.50 (97.92% & 93.09% for Recall and precision, respectively) after held-out tests on a dataset containing 867 images. Overall, the ANPR Pipeline had a whole-plate detection accuracy of 53.73% on raw 67 entry/exit pictures without cropping (36/67; 95% CI: 41.9-65.1%), using fuzzy whole-plate character matching with an 80% threshold. In view of the small number of end-to-end test data, this finding should be seen more as an initial performance estimate than as a statistically reliable indication of deployment-based accuracy for ANPR. The occupancy detection component performed at 97.14% accuracy from 70 test samples manually inspected, while the reservation, transaction log maintenance, and hazard detection capabilities were also demonstrated on the real-world system without quantitative evaluation. These outcomes demonstrate the feasibility of combining localized ANPR technology with low-cost, IoT-enabled parking infrastructure in Nepal, although large-scale implementation should await further end-to-end evaluation and field trials.

Keywords—smart parking, Internet of Things, automatic number plate recognition, Nepali license plates, YOLOv8s, CNN

I. INTRODUCTION

In urban centers of Nepal, vehicle proliferation has outstripped the development of formal parking infrastructure, resulting in

on-street parking and inefficient space utilization. Many of today's parking facilities still use manual logbooks, attendant monitoring, and cash payment methods. This prevents real-time slot availability monitoring, data integrity maintenance, and revenue tracking.

The addition of an element of safety with enclosed and underground parking, to detect possible smoke/fire/gas build-ups without constant human surveillance, will further elevate functionality. Existing ANPR systems, on the other hand, fail to generalize well to the unique layouts and scripts on Nepali license plates [11]. This work proposes a system that combines real-time occupancy monitoring with booking, Nepali-specific ANPR, logging of transactions, and hazard sensing in a single IoT device.

The majority of existing literature on IoT parking focuses on resource management and allocation along with reservation on a network of sensors [1, 2], while higher-level reviews of sensing, communication, and cloud infrastructure for IoT parking are available [3, 4, 13]. Some recent implementations of IoT-enabled parking integrate occupancy sensors with ANPR systems for logging entries and automating parking fee collection [30], some of them incorporating IR sensors along with OCR-based plate recognition for tariff calculation [31], where accuracy has been reported. However, neither of them addresses a specific regional character set or offers a combined solution that integrates hazard detection.

Traditional ALPR pipelines divide identification into localization and recognition modules, as discussed extensively in existing reviews [5, 24]. Recent works employing the family of YOLO object detectors have demonstrated significant improvements in localization [6, 7, 19, 26], and layered localization-segmentation-recognition pipelines are widely effective for ANPR tasks [8, 9, 10]. Regarding Nepali plates, early work [11] employed an SVM classifier for plate recognition, and

Khadka et al. [12] used YOLOv3 alongside a 34-class CNN architecture for character recognition in Nepali.

Instead of introducing new architectures for detection and recognition, this paper adapts and combines them in a low-cost prototype that incorporates reservation-aware sensing, transaction logging, and hazard sensing. It should be noted that this is a prototype-scale study for feasibility validation and not an end-user production system.

Specifically, the prototype integrates real-time slot monitoring and reservations using ESP32/Arduino, IR/ultrasonic sensors, and a Firebase/React web application, along with a Nepali ANPR pipeline that uses YOLOv8s for plate and character detection and a fine-tuned 34-class CNN for character recognition coupled with geometric ordering. Integrated smoke/flame/gas detection is also provided and displayed on a shared dashboard. This paper's contributions are:

- A synchronized workflow combining reservation-aware occupancy sensing, Nepali ANPR, transaction logging, and hazard monitoring with real-time feedback to Firebase is demonstrated on a physical prototype. Both the ANPR pipeline and occupancy sensing are rigorously evaluated quantitatively.
- The challenge of recognizing Nepali license plates with multi-line formatting has been addressed through plate localization, character detection, a fine-tuned 34-class CNN recognizer, and geometric ordering; these are adapted from previous methods [11, 12], representing improvements on an existing framework rather than novel architectural contributions.
- Physical validation with a prototype comprised of four parking bays has been performed. Unlike most reviews, which are primarily qualitative or focus on limited functional subsets (e.g., isolated occupancy sensing, complete ANPR), this study provides a comprehensive evaluation of subsystems and the end-to-end ANPR system (using mAP@0.50, Levenshtein distance, exact-plate accuracy) as well as occupancy detection accuracy. Additionally, the reservation, transaction logging, and hazard-alert functionalities are demonstrated.

II. SYSTEM DESIGN AND ARCHITECTURE

The system combines occupancy sensing using IoT sensors with computer-vision-based image analysis, vehicle identification, cloud synchronization, and environmental hazard monitoring. The data paths between the system components are shown in Figure 1.

The Nepali ANPR engine is responsible for plate/character recognition, localization and character classification based on CNN, rather than general-purpose OCR (see Section III-B).

IR and ultrasonic channels are installed in the sensing layer to monitor parking bays while the cameras located at entry and exit points record images of vehicles, and environmental sensors detect hazardous signs and signals. The Arduino and ESP32 microcontroller nodes receive and send sensor data to the Firebase Realtime Database wirelessly over the internet

[15, 16]. Hazard nodes are additionally connected to SMS alerts using a SIM800L GSM module [17].

The detected entry and exit frames are then sent to the ANPR module, where the first step is to localize the license plate and correct its geometry. Then the license plate image is segmented character by character, followed by classification of those characters using the fine-tuned CNN. After that, the recognized characters are ordered based on their geometric order to create the entire license plate string. This generated plate string and its timestamp are stored in Firebase.

States of occupancy are stored along with the cloud/application layer, vehicle logs, hazard events, fare records, and reservation records. A React web dashboard displays real-time parking availability from Firebase, notifies users of reservations, and controls parking functions. The actual model used in the experiment is a physical prototype. The prototype for system-level testing is shown in Fig. 2.

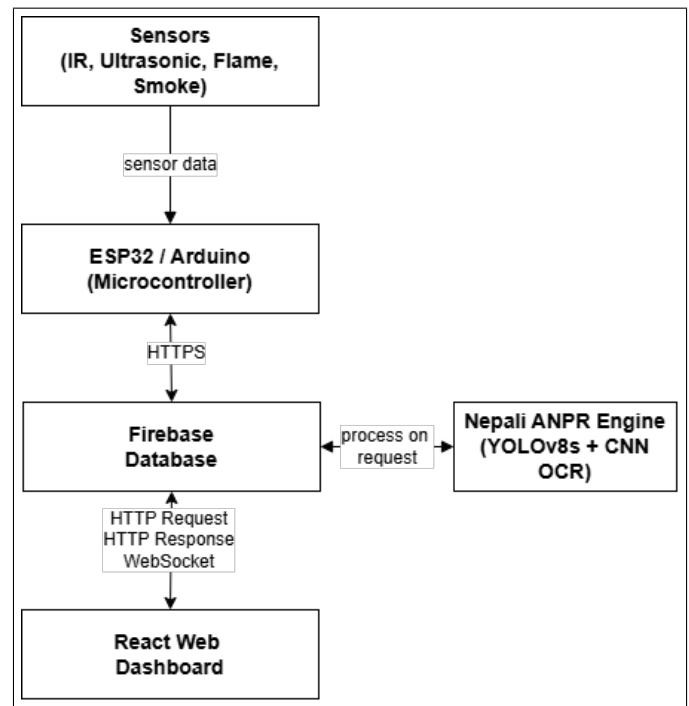


Figure 1: System architecture of the proposed prototype.

The prototype demonstrated in Figure 2 has instrumented parking bays, environmental sensing, and local display units.

III. METHODOLOGICAL FRAMEWORK

A. Occupancy Sensing and Reservation

The prototype uses IR and ultrasonic sensing channels for parking bays. The channel per bay (in use) is determined by the testbed location [22]. Let $S(t) \in \{0, 1\}$ denote a bay's sensor. At any given time t , the output $S(t)$ is 1 if a vehicle is detected. This occupancy state $O(t)$ serves directly as $S(t)$ and is synchronized with Firebase and the dashboard. Reserving a slot from the web app changes the state from "Available" to "Reserved". The number of pending reservations is scanned every five minutes, and each one is

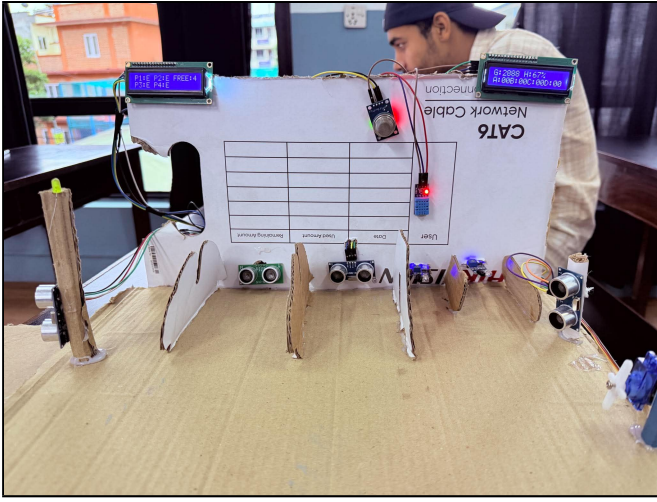


Figure 2: Physical prototype of the system.

given a two-minute confirmation delay before reverting to “Available.”

B. Nepali ANPR and Fare Calculation

The ANPR method is a multi-stage recognition pipeline. Fig. 3 shows the sequence used consistently throughout this work.



Figure 3: Multi-stage Nepali ANPR pipeline.

The YOLOv8s-based plate detector is used to scan the entire frame with a confidence threshold of 0.30. If more than one plate candidate is detected, the box that has the highest level of confidence is maintained. The box selected is cropped from the image with a boundary extended by 3% to decrease edge-character clipping, gray-level conversion,

smoothing and thresholding, and contour-based perspective or rotated-rectangle correction are applied to tilted plates. The two operating values used here are the 0.30 detection threshold and the 0.40 character-localization threshold. These parameters were selected through preliminary trial-and-error experiments rather than through a systematic parameter sweep. This restriction is explored in more detail in Section IV-E.

The second YOLOv8s model, used to crop regions, localizes individual characters on the cropped region. The confidence threshold for each character box is 0.40, below threshold box gets rejected. The boxes that survived are clustered together vertically by the centroid and then ordered horizontally from left to right and fit a multi-line plate formats.

Finally, the recognized character is converted to grayscale, rescaled to 32x32 pixels, and input into a PyTorch CNN (the parameters are specified in Table I), trained on a loss function using the AdamW optimizer [33], and utilizing previously established deep learning and Devanagari character recognition pipelines [20, 21, 25]. The classified characters are then placed geometrically according to their local position to form the final plate string. Errors may occur at multiple stages (plate/character detection, recognition, classification, ordering), and errors were not assigned to stages in the original experiment log. Therefore, we conducted two types of evaluations: plate detector evaluation on 867 images unseen by the plate detector (see Section IV-A) and end-to-end evaluation on the 67 raw images used for evaluation in the end-to-end test set (see Section IV-C).

Table I: CNN Training and Inference Configuration

Parameter	Configuration
CNN input	32 × 32 × 1 grayscale image
Convolution block 1	32 filters + max pooling
Convolution block 2	64 filters + max pooling
Convolution block 3	128 filters + max pooling
Fully connected layer	512 units
Dropout	0.5
CNN output classes	34
Optimizer	AdamW, lr = 0.001
Training loss	CrossEntropyLoss
Training epochs (fine-tuning)	2
Batch size	32
Held-out character-split test accuracy	97.41%
Character-localization threshold	0.40
Plate-detection threshold	0.30
Plate-crop padding	3%

1) Character-Recognition Model and the 34-Class Label Set

Prior to proceeding with fine-tuning a classifier, it seemed appropriate to first benchmark existing generic OCR engines to determine how successful they were when applied in the simplest way. We tried both PaddleOCR [28] and Tesseract [29] in small experiments where these engines were presented with cropped, individual character images extracted from Devanagari plates; in both cases, performance degraded substantially because they had not been trained with small, stylized character shapes such as those on Nepali plates. The character recognition step therefore utilizes a CNN and an

initial set of trained weights provided with the open-source TRaiFIC Nepali ANPR project [27], although they are then fine-tuned on our own labeled character dataset.

However, because we started from externally pre-trained rather than from scratch, a relatively small number of epochs (Table 1) are necessary for the network to reach over 97.4% character classification accuracy on a held-out per-character text split.

However, this indicates that it is other parts of the pipeline, such as plate and individual character localization (e.g., order extraction or translation errors) and the domain shift between the classifier's training images and the data encountered on a specific plate, that are contributing significantly to errors within the system, and no attempt was made to precisely assign the loss value across each of the stages within our system due to the very small end-to-end validation/test set. Since we do not train our character classifier from scratch and because it uses pre-trained models generated by another entity, overlap between the TRaiFIC original dataset and our test set has not been investigated (or audited). This is noted as a limitation in Section IV-E rather than assumed to be absent.

Table II lists the 34 character classes. Rather than decomposing each glyph into a base consonant plus separate vowel-diacritic (mātrā) classes, as a phoneme-level Devanagari OCR system would, this classifier treats each visually distinct glyph cluster – plain consonants (e.g., क, ब, म), consonant-vowel or consonant-consonant conjuncts (e.g., को, प्र, बा, मे), and the ten Devanagari digits – as a single class, matching the granularity at which the character-localization YOLOv8s model already emits bounding boxes. The label set covers only the clusters and digits observed in the training corpus; any cluster absent from it would require retraining to support.

Table II: Character Label Set

Class type	Labels
Consonant / conjunct clusters (24)	क, को, ख, ग, ब, ज, झ, ञ, ट, त, ना, प, प्र, व, बा, भे, म, मे, य, लु, सी, सु, से, ह
Digits (10)	०, १, २, ३, ४, ५, ६, ७, ८, ९

34 character classes total.

Algorithm 1 summarizes recognition and transaction logging.

Let entry and exit timestamps be expressed in seconds. Parking duration in minutes is

$$\Delta t = \frac{t_{\text{out}} - t_{\text{in}}}{60}. \quad (1)$$

For base rate r_b , billing interval m , and incremental rate r_i , the fare is

$$F = \begin{cases} r_b, & \Delta t \leq m, \\ r_b + r_i \left\lceil \frac{\Delta t - m}{m} \right\rceil, & \Delta t > m. \end{cases} \quad (2)$$

C. Evaluation Measures

We report the standard object detection metrics for the plate detector-precision, recall, and mAP at IoU thresholds 0.50

Algorithm 1 ANPR and Fare Logging with Fuzzy Plate Matching

Require: Camera frame I , gate type $g \in \{\text{entry}, \text{exit}\}$

Ensure: Recognized plate string and transaction record

```

1:  $B \leftarrow \text{PlateYOLO}(I)$ 
2: if  $B = \emptyset$  then
3:   return NoPLATEDETECTED
4: end if
5:  $P \leftarrow \text{CropAndCorrect}(I, B)$ 
6:  $C \leftarrow \text{CharacterYOLO}(P)$ 
7:  $\hat{S} \leftarrow \text{ClassifyAndOrder}(C)$ 
8: if  $g = \text{entry}$  then
9:    $t_{\text{in}} \leftarrow \text{CurrentTimestampSeconds}()$ 
10:   $\text{Firebase.Insert}(\hat{S}, t_{\text{in}})$ 
11: else
12:   $R \leftarrow \text{Firebase.GetActiveRecords}()$ 
13:   $M \leftarrow \text{ExactMatch}(\hat{S}, R)$ 
14:  if  $M = \emptyset$  then
15:     $M \leftarrow \text{FuzzyMatch}(\hat{S}, R)$ 
16:    if  $M = \emptyset$  or  $\text{Similarity}(\hat{S}, M.S) < 0.80$  then
17:      return MANUALVERIFICATIONREQUIRED
18:    end if
19:  end if
20:   $t_{\text{in}} \leftarrow M.t_{\text{in}}$ 
21:   $t_{\text{out}} \leftarrow \text{CurrentTimestampSeconds}()$ 
22:   $\Delta t \leftarrow (t_{\text{out}} - t_{\text{in}})/60$ 
23:   $F \leftarrow \text{FareCalculation}(\Delta t)$ 
24:   $\text{Firebase.Update}(M, t_{\text{out}}, F)$ 
25: end if
26: return  $\hat{S}$ 

```

and 0.50:0.95-computed in the standard way [6, 26]. mAP is reported only for the plate-localization detector, and it is evaluated in isolation on the 867-image held-out set described in Section III-E. We evaluate the entire pipeline – plate detection, character localization, classification, and geometric ordering, run end-to-end on raw images – using the following sequence-based metrics, computed on the 67-image end-to-end test set described in Section III-E. For a ground truth string S_i and predicted string $\hat{S}_i$, the per-image normalized Levenshtein similarity (NLS) is

$$\text{NLS}_i = 1 - \frac{\text{EditDist}(S_i, \hat{S}_i)}{\max(|S_i|, |\hat{S}_i|)}. \quad (3)$$

This NLS is a normalized edit-similarity score rather than a standard classification accuracy, and is referred to by the abbreviation *NLS* throughout this paper. The mean NLS reported in Section IV-B is the mean of NLS_i over the 67 end-to-end test images. Because it is calculated over raw images through the complete system (including plate detection), it is a true end-to-end metric, not just a sub-component metric.

The similarity function $\text{Similarity}(\cdot, \cdot)$ invoked during fuzzy plate matching in Algorithm 1 is precisely the normalized Levenshtein similarity of Eq. (3), i.e. $\text{Similarity}(\hat{S}, M.S) = \text{NLS}(\hat{S}, M.S)$.

Apart from being used as an end-to-end evaluation metric, normalized Levenshtein similarity is also used in the operational process of plate matching for entry–exit transactions. In case a prediction does not provide an exact match between the exit and an existing entry record, the predicted plate string is matched with all the stored plates of active records using normalized Levenshtein similarity (Algorithm 1, lines 12–15). The most similar record among the active ones is chosen as the matching candidate. Based on experimental threshold analysis, a value of 0.80 is adopted as the acceptance level for the fuzzy match (Algorithm 1, line 13). If the highest similarity falls below 0.80, the transaction is not linked automatically and is instead flagged for manual verification (MANUALVERIFICATIONREQUIRED, Algorithm 1, line 14).

Per-image precision and recall were derived from the Levenshtein alignment between S_i and $\hat{S}_i$ (matched characters out of the predicted length and ground-truth length, respectively); these are nonstandard sequence alignment metrics that are reported here for thoroughness rather than being presented as exact-match accuracy, and should not be interpreted as CER. CER is not reported because the per-image Levenshtein edit counts required for CER were not captured when running the 67-image end-to-end experiment (see Section IV-E). The *mean* precision, recall, and F1 score were obtained by averaging the respective per-image metrics across the 67 images (treating each image equally). Conversely, the *micro-averaged* values represent results computed by summing counts for all images prior to calculating ratios, i.e., (sum of matched characters)/(sum of predicted characters). They do not have to be the same, and this paper shows both Per-image precision, recall, and F1 were also calculated for the character-localization detector alone and for the classification CNN alone – similar to the sub-component precision, recall, and mAP presented for the plate detector in Table IV – but this data is no longer available from the experiment log and is thus not presented here (see Section IV-E).

D. Environmental Hazard Monitoring

The flame and MQ-series smoke/gas sensors plug into a second microcontroller node. When a defined hazard threshold is triggered, the node updates the dashboard and can send an SMS alert via the SIM800L GSM module. The hazard subsystem was implemented using MQ-series sensors and threshold-based triggering; however, the exact sensor models, calibration procedure, sensing interval, and threshold values were not retained in the experimental documentation. This subsystem has been functionally demonstrated, not numerically measured: no claims are made regarding accuracy of detection or time taken to transmit alerts.

E. Dataset and Experimental Configuration

The plate detection stage was trained in Google Colab using NVIDIA T4 hardware on 8,078 images collected for the “Config Number Plate Detection” dataset [14], subdivided into 5,999 training, 1,212 validation, and 867 test images. The 867-image test portion was retained for evaluating performance in Section IV-A and was not used during training or validation.

For annotation, each image was labeled with an axis-aligned bounding box enclosing the number plate, represented in YOLO format. The training configuration is summarized in Table III, including the use of the AdamW optimizer [33].

The number plate character segmentation model, a YOLOv8s-based model, was trained separately using the “Nepali Vehicles Number Plate Dataset” [18], which contains 2,503 cropped images of Nepali vehicle number plates. Here, all the data are used for training, and specific training, validation, and test counts for this stage were not recorded in the available project logs. Therefore, these quantities are treated as undocumented and would need to be re-established through dataset re-processing or controlled testing before this component can be evaluated reliably, as discussed in Section IV-E.

Because an independent test split was not retained for the character-localization model, no quantitative performance claim is made for this component.

The end-to-end evaluation of the complete ANPR pipeline was conducted on a separate set of 67 unprocessed, ground-truth-annotated entry/exit images, as described in Section IV-B. Unlike the 867-image test set used for the plate detector, these images were neither cropped nor pre-localized. Each image was processed through the complete pipeline, from vehicle number plate detection through character extraction and classification to geometric character ordering, using the same procedure intended for deployment on the physical prototype.

This 67-image dataset is entirely separate from both the plate detection dataset of 8,078 images and the character segmentation training dataset of 2,503 images. It was the only dataset used for the detailed end-to-end evaluation of the complete pipeline. The source of these images, the number of distinct vehicles or number plates represented, the time of image acquisition, camera type and resolution, capture distance and viewing angles, and annotation procedure are not fully documented at present. These limitations are discussed in Section IV-E. Consequently, the relatively small sample size of 67 images should be interpreted as providing preliminary performance estimates rather than a statistically representative evaluation of deployed-system accuracy.

The occupancy-management subsystem was tested manually at 70 parking locations with visually verified availability. In addition, 100 transactions were conducted to validate system synchronization and reservation logging. Parameters such as booking approval rate, gate closure response time, fare-calculation accuracy, reservation synchronization delay, and per-transaction ANPR inference time were not measured during these tests. The subsystem was nevertheless observed to function correctly for the tested scenarios, with these unmeasured performance parameters identified as limitations in Section IV-E.

IV. RESULTS AND DISCUSSION

A. License-Plate Detection Performance

The confusion matrix visualizes the correlation between the Ground-truth annotation and the detector predictions, namely:

Table III: Plate-Detector Training Configuration

Hyperparameter	Value
Backbone	YOLOv8s
Input resolution	640×640
Dataset size (train/val/test)	5,999 / 1,212 / 867
Batch size	4
Epochs (patience)	150 (30)
Optimizer	AdamW
Momentum	0.937
Initial learning rate	0.001
Final LR factor	0.01
Weight decay	0.0005
Warm-up epochs	3.0
NMS IoU threshold	0.7
Box/Cls/DFL loss gain	7.5/0.5/1.5
Augmentation	Mosaic, HSV jitter, rotation, translation, scale, erasing
Random seed	42
Training hardware	NVIDIA T4 (Google Colab)

YOLOv8s backbone.

true positives (correctly detected license plates), false positives (incorrect detections), and false negatives (missed license plates). Using these measures, precision, recall, and F1 - score of the detector can be computed. However, similar to standard classification tasks, for object-detection tasks, there are no well-defined true negatives because the set of non-object regions in an image is not enumeratively given.

Thus, one should not consider the cell located at the bottom-right corner of similar visualizations to represent a standard true-negative count.

Therefore, this Figure represents mainly how the predicted and ground-truth classes are related (but again: be careful about the bottom-right corner cell). Fig 4 shows the confusion-matrix visualization on the hold-out set of 867 images introduced in Section III-E after training the number plate detection model on the training set.

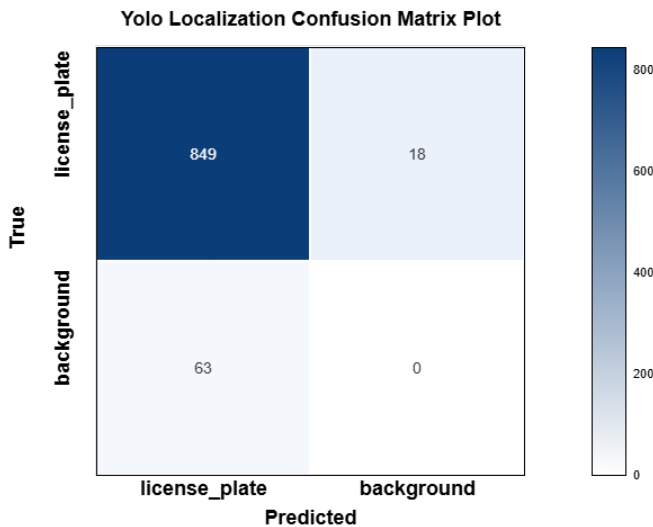


Figure 4: Confusion matrix for license plate localization.

The matrix was evaluated on the 867-image test set.

From Figure 4, the detector achieved a precision value of $849 / (849 + 63) = 93.09\%$ and a recall value of $849 / (849 + 18) = 97.92\%$ yielding an F1-score of 95.45%. It is observed

that the detector missed a relatively low number of ground truths, while some amount of false positives were detected in some non-plate regions. The mAP@0.50 and mAP@0.50:0.95 were derived from applying the trained YOLOv8s model on the same hold-out 867-image test set.

Since this test set was not involved in training/validating the models, it doesn't have to match the precision and recall calculations from the confusion matrix arithmetically.

Additionally, the mAP@0.50:0.95 is lower than mAP@0.50 since this calculation averages all IoU values from 0.5 to 0.95 instead of only 0.5.

Table IV: License-Plate Localization Performance

Metric	Value
mAP@0.50	95.44%
mAP@0.50:0.95	80.07%
Precision	93.09%
Recall	97.92%
F1-score	95.45%

YOLOv8s-based detector, evaluated on the 867-image test set.

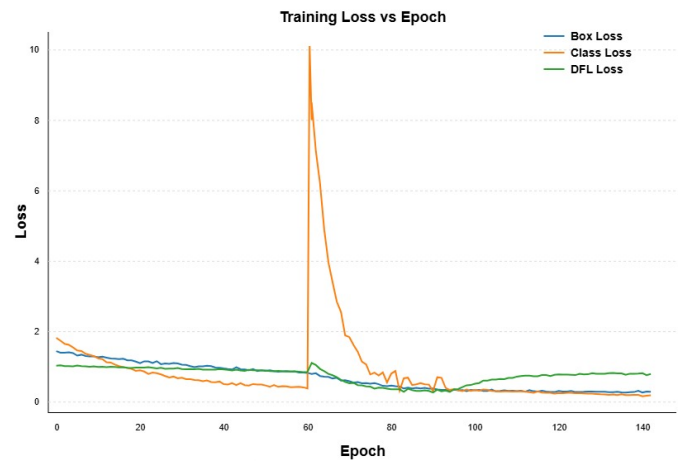


Figure 5: Epoch vs. loss for the YOLOv8s localizer.

B. End-to-End ANPR Performance

Table V: End-to-End ANPR Performance

Metric	Value
Exact whole-plate accuracy (36/67)	53.73%
95% Wilson CI	41.9%–65.1%
Mean NLS (normalized Levenshtein similarity)	91.48%
Mean precision (Levenshtein-based)	91.73%
Mean recall (Levenshtein-based)	93.88%
Mean F1 (Levenshtein-based)	92.69%
Micro precision	88.39%
Micro recall	90.00%
Micro F1	89.19%

Evaluated on 67 end-to-end test images. Exact whole-plate accuracy is the primary performance measure; NLS- and Levenshtein-based metrics provide partial-credit measures of prediction similarity and are not directly comparable to detector mAP. Inference throughput was not measured (Section IV-E).

The full pipeline-which consisted of the plate detector, character localization, CNN, and geometric ordering-was evaluated end-to-end on the independent 67 raw test images in Sec. III-E. These 67 images are distinct from the 867 images used

in Sec. IV-A when the plate detector itself was evaluated independently, and they were processed directly without the use of any human correction to the detection/localization steps before processing them, meaning the final metrics should capture the end-to-end pipeline's errors at each of the steps. The overall end-to-end performance of our full pipeline resulted in an exact whole-plate accuracy of 53.73% or 36 out of 67 images (95% Wilson CI: 41.9%–65.1%).

Our end-to-end precision, recall, F1-score, and Levenshtein similarity can be found in Table V.

The whole-plate accuracy metric is the central ANPR end-to-end metric of interest, whereas the Levenshtein metrics provide some credit for estimates that are somewhat but not completely accurate (these metrics are not identical to traditional mAP for detectors and should not be compared to exact match accuracy). Based on our relatively small end-to-end test set, we should be particularly cautious when interpreting this number and intend to evaluate on larger, unbiased data in the near future, as, for example, one incorrect item out of 67 affects the accuracy by nearly 1.5 points. It is not possible to report the specific precision, recall, or F1 scores on any specific stage, as the metrics are aggregated across the entire end-to-end pipeline; our held-out per-character test classification accuracy, for example, was 97.41%, which could explain why errors in class prediction or recognition alone do not entirely make up for our end-to-end error.

The mean NLS and mean Levenshtein-based recall were calculated as:

$$\text{Mean NLS} = \frac{1}{N} \sum_{i=1}^N \text{NLS}_i \quad (4)$$

$$\text{Mean Recall} = \frac{1}{N} \sum_{i=1}^N R_i \quad (5)$$

where $N = 67$ denotes the total number of raw images in the end-to-end test set, NLS_i is the normalized Levenshtein similarity for the i -th image, and R_i is the Levenshtein-based recall for the i -th test sample. The corresponding micro-averaged precision, recall, and F1-score are computed by pooling the matched, predicted, and ground-truth character counts across all 67 images before calculating a single ratio, rather than averaging the 67 per-image ratios. Thus, the macro- and micro-averaged measures are not identical and provide complementary information about end-to-end recognition performance.

Displays normalized ground-truth and predicted Nepali license-plate strings and their exact-match results.

We can observe that since the exact match result is only 53.73% and the average NLS is 91.48%, some errors are only partial character insertions, deletions, or substitutions of a few characters. Levenshtein-based similarity measure is useful for estimating the severity of partial matches, whereas an exact match accuracy score tells how long the whole plate was correctly identified[32]. This distinction is important in a deployed parking system: a near-miss prediction could cause a transaction to be linked to the wrong vehicle record, for

Representative Nepali ANPR Test Cases

Exact_Match: 1 = exact recognition, 0 = mismatch

Test_ID	Actual_Normalized	Predicted_Normalized	Exact_Match
Test_0001	बागमतीप्रदेश०२०४३५०९	यबाझचतुप्र०२०४३५०९	0
Test_0008	गशय५३८	गशय५३८	1
Test_0027	बा१५च३८९७	लुबा१५च३८९७	0
Test_0048	ना४३पर८९६	ना४३पर८९६	1
Test_0065	बा३च८०८४	बा३च८०८४	1

Figure 6: Representative end-to-end ANPR test cases showing ground-truth and predicted Nepali license-plate strings and their exact-match outcomes.

example matching plate 'x' to a stored record for a different vehicle 'y' instead of the correct record. The NLS values obtained across varying similarity scores are therefore used to find a threshold value suitable for fuzzy matching of entry and exit reads (see Fig. 7 below).

Actual_Normalized	Predicted_Normalized	Fuzzy_Score_Percent	Match_Type
बागमतीप्रदेश०२०४३५०९	यबाझचतुप्र०२०४३५०९	68.18	NO MATCH
गशय५३८	गशय५३८	100.00	EXACT MATCH
गशय६३०	गशय६३०	100.00	EXACT MATCH
बा१५च३८९७	बा१५च३८९७	100.00	EXACT MATCH
बा३च८०८४	लुबा१५च३८९७	90.00	FUZZY MATCH

Figure 7: Corrected fuzzy matching results for sample number plates, comparing actual and predicted normalized values with threshold value 0.80.

This idea of distinguishing perfect accuracy from normalized Levenshtein similarity underlies the fuzzy approach of comparing the entry and exit reads. Though an extra/incorrect character inserted in the input could be the reason why a perfect database search failed, this match with a few character differences could be the actual corresponding entry/exit for the actual active vehicle. A perfect database search is thus tried out, and only if it does not produce any results will normalized Levenshtein similarity be tried and used for comparisons. Using a threshold value established through experimental evaluation of different similarity threshold values, summarized in Figure7, we use a match threshold of 0.80; any similarity score below this threshold is not automatically accepted as a valid match.

The ANPR subsystem saves recognized vehicle plate strings along with entry and exit timestamps and related fare records in the database. These records can be seen as vehicle usage information, which brings up privacy concerns when the system is used in real-world settings. If someone gains access to these records, they could find out about vehicle movements and parking behavior. That is why a production version must use authentication and access controls. It must also

protect data both when it is stored and when it is sent over networks. A clear data-retention policy should be in place to say how long plate numbers, timestamps, and transaction records are kept, based on what's needed for operations and billing. Once records are no longer needed, they should be securely deleted or made anonymous. The current prototype is focused on testing the system's functions and does not include a formal privacy policy or data-retention plan. Therefore, these measures are considered requirements for when the system is deployed in the future.

C. Parking-Slot Occupancy Detection Performance

The parking testbed was tested with a corpus of 70 manually annotated parking instances. There were 68 correct results, for a final accuracy of 97.14% (95% Wilson score interval around 90.2%–99.2%). The confusion matrix can be observed in Table VI. The wide confidence interval is likely a result of the small number of test instances and should be taken into account when comparing results.

Table VI: Parking-Slot Detection Results

	Pred. Occupied	Pred. Empty
Actual Occupied	34	1
Actual Empty	1	34

Based on 70 manually verified test cases.

The confusion matrix is symmetric, with one false-empty case and one false-occupied case out of 70, giving an accuracy of $68/70 = 97.14\%$.

D. Reservation, Synchronization, Fare Logging, and Hazards

While utilizing the reservation and fare logging workflows, the prototype successfully synchronized occupancy and transaction information between Firebase and the dashboard through more than one hundred transactions, with no apparent data loss detected during each process trial. We accordingly state that this functionality is *functionally demonstrated*, not *validated*: reservation and gate success rates, fare-calculation accuracy, synchronization latency, and trial ANPR durations were not measured, and do not persist in the logs that we retained (Section IV-E). Similarly, the smoke/flame/gas subsystem is presented only as functionally integrated, not as a tested quantitative sensor device.

E. Limitations

Data availability: Train/validation/test counts and training logs for the character localization stage were not stored, nor were reservation success, gate response latency, fare calculation accuracy, ANPR latency, and synchronization latency. No values are filled in for these absences; experiments should use a standard, documented logging method.

Recognition Accuracy: Our evaluation in §IV-B is end-to-end on raw image data, so results account for plate localization errors. Nevertheless, the testing set of 67 images is modest, and a single additional correct or incorrect prediction changes the accuracy by approximately 1.5 percentage points of accuracy for both measures reported below. So, these should be taken as indicative.

Our results may not reflect performance with dirty, bent, embossed, or occluded plates, and under extreme lighting (either low or backlit). No 34-character-class confusion matrix was retained or per-stage results cataloged, so we are not in a position to diagnose specific character confusion. The character classifier was based on fine-tuning external TRaFiC weights [27]. We did not audit the extent to which its training/tuning data overlapped with the 67 images used for testing.

Although the 0.80 similarity threshold was selected experimentally for approximate entry–exit matching, it should be further validated on a larger set of independent entry–exit vehicle pairs to quantify false-match and missed-match rates.

Scope and hyperparameters for Deployment: The occupancy subsystem only uses a 4-bay model, connected via Wi-Fi, and not multi-floor or other forms of broader deployment. Neither the hazard's detection sensitivity, false positive rates, nor alertness latency were evaluated. The exact models of the MQ sensor, calibration steps, as well as threshold decisions, are still unspecified.

The thresholds of 0.30 for plate detection, 0.40 for char locator, and 3% crop were also chosen by hand-tuned experimentation rather than by exhaustively logging results from testing different possible threshold combinations.

ANPR latencies per trial and details on the deployment hardware were not recorded in any logging system. Plate strings, vehicle timestamp logs, and logs of their fare could be used as vehicle usage records; however, security such as at-rest encryption and data retention/access policies should first be considered.

F. Future Work

Our future work includes expanding the character-recognition dataset, such as by using plates from more locations that are more varied in terms of degree of degradation and time of day (including nighttime). For end-to-end evaluation, we should scale up beyond the 67-image test set using a larger, independently sampled dataset, which should be subjected to a stage-wise analysis of errors of individual stages (plate detection and character detection/recognition/ordering) along with a confusion matrix for all 34 character classes. Finally, we should test stage-wise effects of detection/localization thresholds and crop-padding ratios on the end-to-end system via ablation studies.

For the complete system, a larger multi-floor pilot run is needed to gauge network dependability, reservation success rates, gate opening delays, synchronization issues, fare logging performance, and hazard alerting performance over a fixed test plan.

Additionally, recover and explicitly document the lost training/validation/test split for the character-localization algorithm and estimate the basic inference latency of the character-localization algorithm, as discussed in Sec. IV-E.

V. CONCLUSION

This paper presented a smart parking prototype that included parking space detection and booking, a YOLOv8s-based detec-

tion pipeline and CNN-based character recognition for Nepali automatic number plate recognition (ANPR), logging of entry and exit, logging of fare, and finally environmental hazard detection using a Firebase-based system. Instead of developing an entirely new detection or recognition framework, this system applies adapted ANPR methods in Nepali [11, 12] on low-cost IoT devices with a comprehensive parking-management platform.

There are three key findings from our evaluations. First, the four subsystem components have been integrated and work correctly with low-cost technology (e.g., ESP32/Arduino devices, Firebase, and React dashboard).

Second, the plate detector performance had 95.44% mAP@0.50 and 95.45% F1-Score on an independent test of 867 images.

However, for performance relevant for an operational system, end-to-end evaluation on 67 raw images gave an exact match whole-plate accuracy of only 53.73% (36/67; 95% CI: 41.9% - 65.1%); although for NLS it was still 91.48%. This exact match is therefore the relevant indicator of operational ANPR accuracy and should be considered preliminary given the small test size. Thirdly, the occupancy detection component can be seen to identify 68 of 70 manually validated images (accuracy 97.14%). The reservation, transaction, and hazard logging subsystems worked as prototype functionality but were not measured for success rate, latency, and accuracy.

To improve transaction linkage when exact plate recognition fails, the system additionally employs fuzzy plate matching against active parking records using an experimentally selected similarity threshold of 0.80, with low-confidence matches referred for manual verification.

In essence, this paper has demonstrated the feasibility of a Nepali smart parking solution combining occupancy detection, ANPR, transaction logging, and hazard detection with low-cost components in a single integrated prototype. The disparity between plate-detector performance and the 53.73% end-to-end exact-match rate indicates that the ANPR stage, apart from plate detection, has issues, and individual error sources contributing to this remain undetermined. Large, independent, end-to-end and occupancy detection tests are needed before deployment, with further benchmarks included for reservation success rate, gate control reliability, fare correctness, synchronization speed between stages, and hazard-detection efficacy. This highlights the contribution as application-specific integration and testing, rather than algorithm design, and contributes to closely related prior work [30, 31].

REFERENCES

- [1] Y. Geng and C. G. Cassandras, "New 'smart parking' system based on resource allocation and reservations," *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 3, pp. 1129–1139, Sep. 2013, doi: 10.1109/TITS.2013.2252428.
- [2] W. Alsafery, B. Alturki, S. Reiff-Marganiec, and K. Jambri, "Smart car parking system solution for the Internet of Things in smart cities," in *Proc. 1st Int. Conf. Comput. Appl. Inf. Security (ICCAIS)*, 2018, pp. 1–5, doi: 10.1109/CAIS.2018.8442004.
- [3] M. G. Diaz Ogas, R. Fabregat, and S. Aciar, "Survey of smart parking systems," *Appl. Sci.*, vol. 10, no. 11, Art. no. 3872, 2020, doi: 10.3390/app10113872.
- [4] A. Fahim, M. Hasan, and M. A. Chowdhury, "Smart parking systems: Comprehensive review based on various aspects," *Heliyon*, vol. 7, no. 5, Art. no. e07050, 2021, doi: 10.1016/j.heliyon.2021.e07050.
- [5] S. Du, M. Ibrahim, M. Shehata, and W. Badawy, "Automatic license plate recognition (ALPR): A state-of-the-art review," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 2, pp. 311–325, Feb. 2013, doi: 10.1109/TCSVT.2012.2203741.
- [6] J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.
- [7] G. Jocher, A. Chaurasia, and J. Qiu, "Ultralytics YOLO," 2023. [Online]. Available: github.com/ultralytics/ultralytics.
- [8] S. M. Silva and C. R. Jung, "License plate detection and recognition in unconstrained scenarios," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 580–596.
- [9] Z. Xu, W. Yang, A. Meng, N. Lu, H. Huang, C. Ying, and L. Huang, "Towards end-to-end license plate detection and recognition: A large dataset and baseline," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 255–271.
- [10] R. Laroca, E. Severo, L. A. Zanlorensi, L. S. Oliveira, G. R. Goncalves, W. R. Schwartz, and D. Menotti, "A robust real-time automatic license plate recognition based on the YOLO detector," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2018, pp. 1–10, doi: 10.1109/IJCNN.2018.8489629.
- [11] A. K. Pant, P. K. Gyawali, and S. Acharya, "Automatic Nepali number plate recognition with support vector machines," in *Proc. 9th Int. Conf. Software, Knowledge, Information Management and Applications (SKIMA)*, 2015, pp. 92–99.
- [12] S. Khadka, S. Baral, S. Tiwari, and S. K. Ghimire, "Character recognition of Nepali number plate," *arXiv preprint arXiv:2606.28946*, 2026.
- [13] F. Al-Turjman and A. Malekloo, "Smart parking in IoT-enabled cities: A survey," *Sustain. Cities Soc.*, vol. 49, Art. no. 101608, 2019, doi: 10.1016/j.scs.2019.101608.
- [14] I. Subedi, "Config number plate detection," Kaggle dataset, 2023. [Online]. Available: <https://www.kaggle.com/datasets/ishworsubedi/config-number-plate-detection> (accessed 2026).
- [15] Espressif Systems, "ESP32 series datasheet." [Online]. Available: documentation.espressif.com/esp32_datasheet_en.pdf.
- [16] Google, "Firebase Realtime Database documentation." [Online]. Available: firebase.google.com/docs/database.
- [17] SIMCom Wireless Solutions, "SIM800L product specification and hardware documentation." [Online]. Available: simcom.com/product/SIM800.html.
- [18] Inspiring Lab, "Nepali vehicles number plate dataset," Kaggle dataset, 2024. [Online]. Available: <https://www.kaggle.com/datasets/inspiring-lab/nepali-vehicles-number-plate-dataset> (accessed 2026).
- [19] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020.
- [20] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, 2014.
- [21] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.
- [22] S. Mondal et al., "Smart parking system using Arduino and ultrasonic sensors," *Int. J. Innov. Eng. Sci. (IJIES)*, vol. 12, no. 9, pp. 22–28, Sep. 2025, doi: 10.35940/ijies.E1102.12090925.
- [23] M. Altamash, M. Rayyan, S. A. Khan, and H. Parveen, "Intelligent IoT-based fire and gas leakage detection system:

- Integrating sensor fusion, edge AI, and visual verification,” *Int. J. Eng. Res. Technol. (IJERT)*, vol. 15, no. 4, Apr. 2026.
- [24] Lubna, N. Mufti, and S. A. A. Shah, “Automatic number plate recognition: A detailed survey of relevant algorithms,” *Sensors*, vol. 21, no. 9, Art. no. 3028, 2021, doi: 10.3390/s21093028.
 - [25] I. Dokare, S. Gadage, K. Kharde, S. Bhare, and R. Jadhav, “Recognition of handwritten Devanagari character using convolutional neural network,” in *Proc. 3rd Int. Conf. Signal Process. Commun. (ICSPC)*, 2021, pp. 353–359, doi: 10.1109/ICSPC51351.2021.9451716.
 - [26] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You only look once: Unified, real-time object detection,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, 2016, pp. 779–788, doi: 10.1109/CVPR.2016.91.
 - [27] sanzgrapher, “TRaiFIC: Automatic Nepali number plate recognition system,” GitHub repository, 2026. [Online]. Available: github.com/sanzgrapher/TRaiFIC-ANPR-Nepali-Number-Plate-Detection-plus-Character-Recognition (accessed 2026).
 - [28] PaddlePaddle, “PaddleOCR: Awesome multilingual OCR toolkits based on PaddlePaddle,” GitHub repository. [Online]. Available: github.com/PaddlePaddle/PaddleOCR.
 - [29] R. Smith, “An overview of the Tesseract OCR engine,” in *Proc. 9th Int. Conf. Document Anal. Recognit. (ICDAR)*, 2007, pp. 629–633, doi: 10.1109/ICDAR.2007.4376991.
 - [30] A. Ditta, M. M. Ahmed, T. Mazhar, T. Shahzad, Y. Alahmed, and H. Hamam, “Number plate recognition smart parking management system using IoT,” *Measurement: Sensors*, vol. 37, Art. no. 101409, 2025, doi: 10.1016/j.measen.2024.101409.
 - [31] G. Pradhan, M. R. Prusty, V. S. Negi, and S. Chinara, “Advanced IoT-integrated parking systems with automated license plate recognition and payment management,” *Scientific Reports*, vol. 15, Art. no. 2388, Jan. 2025, doi: 10.1038/s41598-025-86441-w.
 - [32] F. M. Oliveira-Neto, L. D. Han, and M. K. Jeong, “An online self-learning algorithm for license plate matching,” *IEEE Trans. Intell. Transp. Syst.*, vol. 14, no. 4, pp. 1806–1816, Dec. 2013, doi: 10.1109/TITS.2013.2270107.
 - [33] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2019.