Real-Time Parking Slot Detection and Analytics Using YOLOv11n and MongoDB

Dr. Gayathri Devi S
School of Computer Science and Engineering, Vellore Institute of Technology, Chennai
Vellore Institute of Technology
Chennai, India
gayathridevi.s@vit.ac.in

Anirudh R
School of Computer Science and Engineering, Vellore Institute of Technology, Chennai
Vellore Institute of Technology
Chennai, India
anirudh.r2022@vitstudent.ac.in

Inesh ST
School of Computer Science and Engineering, Vellore Institute of Technology, Chennai
Vellore Institute of Technology
Chennai, India
inesh.st2022@vitstudent.ac.in

***Abstract*—Colleges and other institutions in cities are experiencing congestions in the parking lots prompting drivers to squander on finding empty spaces. The current paper introduces a Smart Parking Availability Detection System based on YOLOv11n and visualizes the free and occupied parking spots automatically. The model was not trained on direct CCTV access and was trained using publicly available parking datasets and tested on recorded videos which were simulating real-time input. The system conducts frame-by-frame analysis after every two seconds, categorizes parking slots as busy or empty and stores the results to a MongoDB database in NoSQL format with the timestamps and slot numbers. The accumulated data will provide log based analytics like prediction of peak hour and trend of use in the future. On car and free-space classes, YOLOv11n obtained 0.932, recall of 0.861, and mAP 50 of 0.905, which is reliable generalization. The system architecture is based on the integration of YOLOv11n inference with FastAPI backend, MongoDB logging and a web-based dashboard to visualize slot occupancy in near real-time.**

***Keywords*—YOLOv11n, Smart Parking, Object Detection, MongoDB, Real-time Analytics, NoSQL.**

I. Introduction

The rapid increase in vehicle ownership across metropolitan and institutional environments has intensified parking challenges, resulting in congestion, wasted fuel, and driver frustration [1]. Parking control thus became one of the primary aspects of the contemporary Smart City projects, which lay stress on the application of real-time analytics and smart transportation systems to enhance the movement of people in the city.

Traditional parking systems that use sensors, whilst accurate, involve expensive hardware efforts and maintenance. Conversely, computer vision (CV) and deep-learning solutions have been proposed as scalable and affordably priced substitutes to occupancy of parking spots based on visual data. The YOLO (You Only Look Once) line of object detectors offers the most recent accuracy and inference rate, which makes it a perfect fit in dynamic parking settings and in scenarios of continuous monitoring.

Nevertheless, initiating such vision-based systems poses mass unstructured data of high frequency which cannot be effectively processed in a traditional relational database. NoSQL databases, especially MongoDB, have flexible schema-design, horizontal-scalability, and built-in-time-series, which makes them a good fit to store frame-level detections and analytics in real time [2][3].

This study offers an object-detected, NoSQL-based smart parking detection framework including YOLOv11n-based object detection and streamlined MongoDB storage and analytics. It is built with a combination of detection, data management on the backend, and visualization to provide near real-time slot

availability data using a web interface. The system was also tested on recorded footage of parking-lots despite the absence of live CCTV feeds because the development required simulating the operation of the system. The proposed architecture will facilitate a smooth process of integrating live cameras and provide scalable, data-driven parking analytics in the future to implement smart campuses and cities.

II. Literature review

A. YOLO-Based Object Detection for Parking Systems

Redmon et al. [1] introduced the YOLO (You Only Look Once) framework as a single-stage real-time object detector, offering superior speed compared to region-based approaches. Successive advancements, YOLOv8 through YOLOv11n, have significantly improved detection accuracy, model compression, and scalability for embedded deployment.

da Luz et al. [2] conducted a comparative evaluation of YOLOv8–YOLOv11n for vehicle detection using pixel-wise region-of-interest (ROI) selection, demonstrating YOLOv11n's superior precision and inference stability. Jiang et al. [3] proposed a two-stage parking space detection architecture utilizing YOLOv11n and geometric post-filtering to enhance bounding box localization.

While these studies achieve high accuracy on standard datasets such as CNRPark and PKLot, they primarily focus on model-level performance without extending to real-time data storage or analytics integration.

B. Computer Vision in Smart Parking and Urban Mobility

Vlahogianni et al. [4] developed a predictive model for real-time parking availability based on urban mobility data, emphasizing the integration of spatial-temporal features. Grbić and Koch (2023) applied deep learning to classify parking occupancy on the CNRPark-EXT dataset, addressing illumination and occlusion challenges.

Yuldashev et al. (2023) employed lightweight MobileNetV3 for per-slot classification in low-power IoT devices, while Nguyen and Sartipi (2024) explored unsupervised parking spot segmentation to minimize manual labeling costs. These studies established strong baselines for vision-driven occupancy detection but did not consider scalable backend or temporal data management components necessary for smart infrastructure analytics.

C. NoSQL Databases for Real-Time Video Analytics

Cooper et al. [5] introduced the YCSB benchmark, establishing a framework for evaluating NoSQL systems under various read–write workloads. Abramova et al. [6] conducted a comparative study of MongoDB, Cassandra, and HBase, emphasizing their trade-offs in latency and horizontal scalability.

Davoudian et al. [7] analyzed NoSQL data models for distributed and time-series storage, highlighting MongoDB's adaptability for real-time event ingestion. In the context of smart infrastructure, Chao et al. (2024) demonstrated MongoDB's suitability for handling high-frequency IoT telemetry streams.

However, prior works did not integrate computer vision pipelines with MongoDB for managing frame-based inference outputs or employ index-optimized schemas for time-stamped detection events.

D. End-to-End Smart Parking Architectures

Rizvi et al. [8] designed an IoT-based smart parking prototype utilizing ultrasonic sensors and cloud dashboards, demonstrating the feasibility of real-time slot monitoring. Ahmed et al. [9] implemented a vision-based parking system using YOLOv11n and OpenCV for binary occupancy detection.

While these architectures showcase functional prototypes, they largely lack NoSQL-based data persistence and do not perform continuous inference logging, limiting their use in long-term analytics and trend forecasting.

E. Identified Research Gaps

Existing research demonstrates robust YOLO-based parking detection pipelines and advances in real-time NoSQL systems but fails to unify both domains.

Specifically:

1. Integration of YOLOv11n-based detection with NoSQL (MongoDB) for

real-time analytics remains underexplored.

2. Schema and indexing strategies tailored for frame-wise object detection logs are rarely evaluated.

3. Current studies neglect temporal aggregation and occupancy trend prediction using database-stored logs.

4. Most implementations test detection accuracy offline rather than within end-to-end real-time pipelines involving API communication and database updates.

The proposed system addresses these gaps by combining YOLOv11n for object detection with MongoDB-based storage and analytics, forming a fully integrated, near real-time Smart Parking infrastructure.

III. Proposed Methodology

A. System Architecture

Fig. 1 presents the general workflow of the proposed Smart Parking System based on the use of the YOLOv11nn. The design is modular with the integration of the detection layer, the backend layer, the database layer and the visualization layer to run real-time efficient processing.

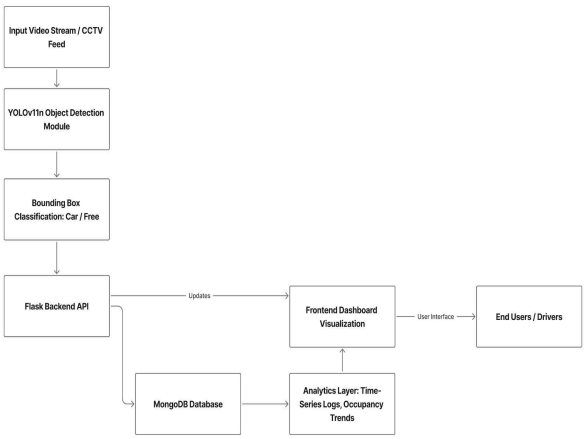

Fig. 1. *System architecture of the proposed YOLOv11n-based Smart Parking System.*

The system starts with an input video stream which is obtained either through a CCTV or a simulated channel. Frames are fed to the YOLOv11n Object Detection Module which does a single stage detection task and classifies each parking slot as an occupied or free car. The processed detections are subsequently transmitted through Flask backend API that enables the REST based communication between the inference engine and the database.

Each detection event, with its time and its confidence level, is recorded in the MongoDB database, making it possible to implement a flexible NoSQL document-based database. On top of this, queries of occupancy trends, time-series logs and long-term parking statistics are supported by an analytics layer. At the same time, the Flask API transmits updates to the frontend dashboard-based visualization every 2 seconds, allowing users the color-coded (free/occupied) interface to the visualization. Lastly, the system provides real-time availability to end users and drivers, and completes a real-time feedback process between identification and effective visualization.

The layered architecture provides high throughput, scalability and low latency which is very appropriate in the deployment of smart-campus and urban parking.

B. Dataset Preparation

The model was trained using Roboflow Parking datasets, comprising various lighting and weather conditions such as sunny, overcast, and rainy environments. Images were annotated in YOLO format with two class labels: *car* (occupied) and *free* (vacant).

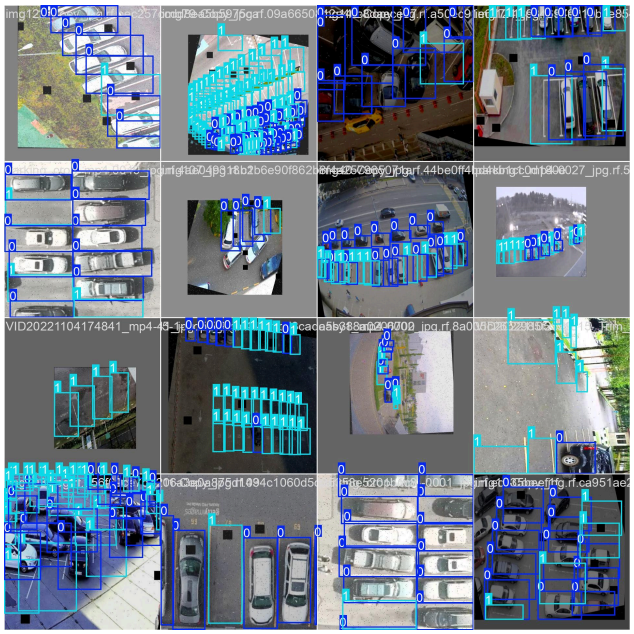

Fig. 2. *Sample annotated training images showing labeled parking slots for "car" (occupied) and "free" (vacant) classes from the CNRPark-EXT dataset.*

Fig. 3. *Validation set predictions demonstrating model generalization under varied lighting and viewpoint conditions.*

The final dataset contained approximately 30,000 annotated instances split into *car* and *free* classes.

C. Model Training

YOLOv11n was trained for 100 epochs on Google Colab (Tesla T4 GPU).

The training and validation loss curves shown in *Fig. 4* illustrate consistent convergence, with gradual reduction in box and classification losses and no signs of overfitting.

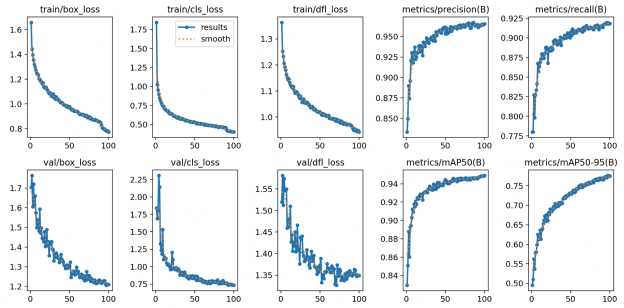

Fig. 4. *Training and validation loss and performance metrics across 100 epochs for YOLOv11n.*

The precision, recall, and mean Average Precision (mAP) metrics improved steadily over the epochs, confirming robust generalization across both training and validation splits.

D. Near Real-Time Frame Processing

The video feed for testing was processed every two seconds, performing inference and updating the MongoDB database with occupancy status. This interval balances computational efficiency with near real-time response, suitable for smart parking where occupancy changes infrequently.

E. MongoDB Integration

Each inference cycle inserts a document:


```
{
  "_id": { "$oid": "69033b153e7ed3865972125e" },
  "timestamp": "2025-10-30T15:46:53.450229",
  "status": {
    "spot_1": "occupied",
    "spot_2": "occupied",
    "spot_3": "occupied",
    ...
    "spot_21": "free",
    "spot_31": "free",
  ...
  }
}
```


Indexes on slot_id and timestamp enable fast query aggregation for future analytics (e.g., hourly occupancy rate, peak usage).

F. Frontend Visualization

A Flask dashboard displays live parking-lot status. Each slot is represented as a colored rectangle-green for free, red for occupied, and updated dynamically from MongoDB queries.

IV. Results and Discussion

A. Model Performance

YOLOv11n model was tested based on common performance measurements such as Precision, Recall, mean Average Precision (mAP) and F1-score. The analysis of the loss-curve and the steady inclination of the precision and the recall values confirmed that after 100 epochs of training, the model converged well with little overfitting.

Table I.

YOLOv11n Model Performance Metrics

Metric	Training Results	Validation Results
Precision	0.965	0.932
Recall	0.918	0.862
mAP@50	0.949	0.905
mAP@50–95	0.776	0.613

As depicted in Fig. 4, the training and validation losses decreased steadily with the number of epochs whereas the precision and recall curves showed a monotonic improvement, which suggests that learning is stable, and the generalization is good. The results of the final quantitative measurements are summarized in Table I, and with YOLOv11n achieving the mAP@50 of 0.905 and mAP@50-95 of 0.613 on the validation sample.

The performance in the classes is represented in the Precision-Recall and Confidence-F1 dependencies (Figs. 5-8). The optimum balance of precision and recall can be shown by the high F1-score of 0.94 at the confidence threshold of 0.46. The trends of both the car and the free classes were similar and this proved that the model is consistent across categories.

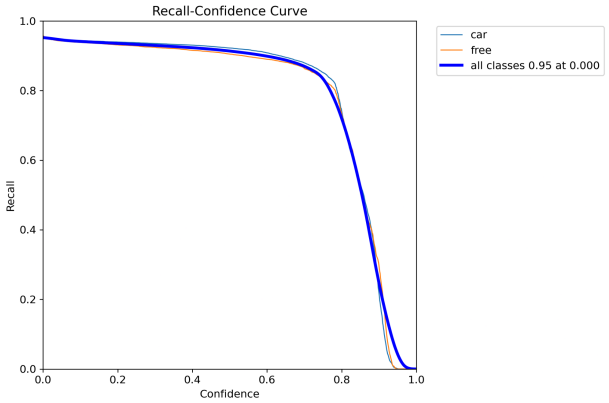

Fig. 5. *Recall–Confidence curve illustrating stable recall up to 0.8 confidence threshold for both "car" and "free" classes.*

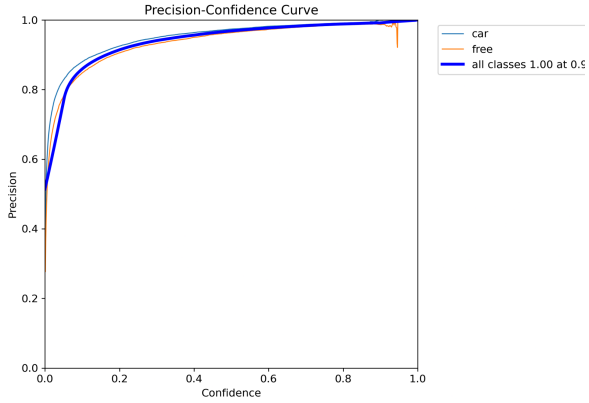

Fig. 6. *Precision–Confidence curve demonstrating consistent precision above 0.9 across confidence levels.*

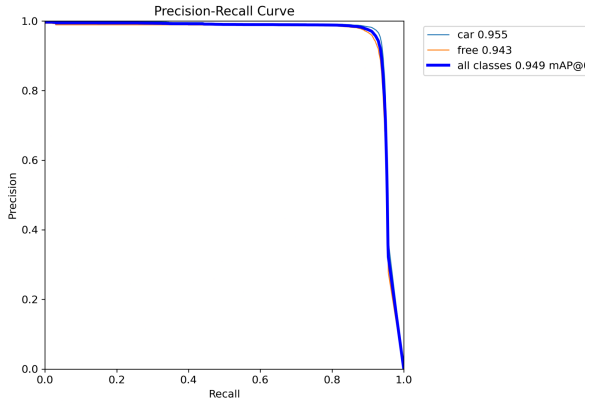

Fig. 7. *Precision–Recall curve showing mAP@50 = 0.949 for all classes, confirming high detection reliability.*

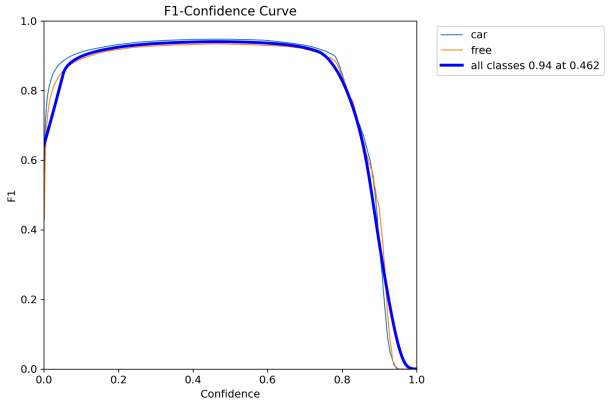

Fig. 8. *F1–Confidence curve showing a peak F1 of 0.94 at confidence 0.462, indicating optimal threshold balance.*

The confusion matrices (Figs. 9 and 10) also confirm the reliability of classification with more than 93-94 percent accuracy in both classes and few false positives in the partial occlusion conditions.

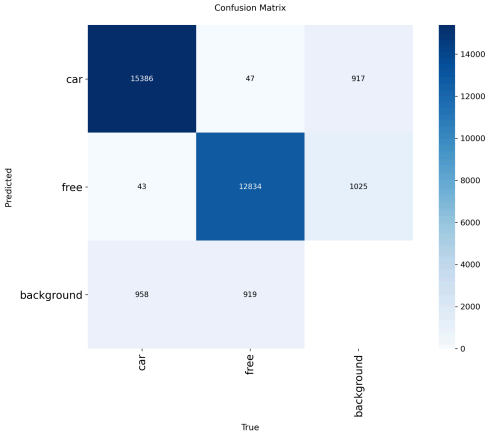

Fig. 9. *Confusion matrix representing raw detection counts for "car," "free," and background categories.*

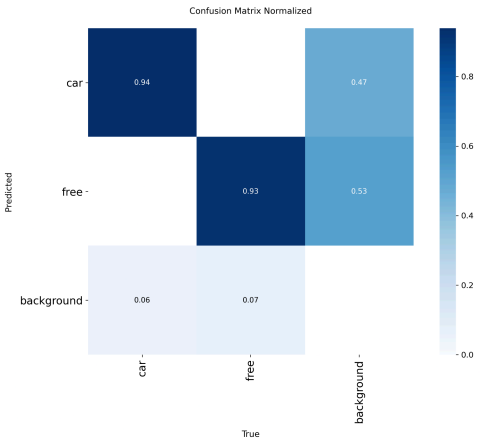

Fig. 10. *Normalized confusion matrix showing 93–94 % correct classification with minimal false predictions.*

Overall, the model could real-time infer about 2.6 ms per frame (≈ 380 FPS on a Tesla T4 GPU), which is enough to implement it in real-time parking surveillance systems. The findings support the fact that YOLOv11n offers a decent trade-off between accuracy and computational efficiency in detecting parking-lot occupancy.

B. Detection Visualization

The YOLOv11n model demonstrated robust detection capabilities across diverse parking layouts and lighting conditions.

Fig. 11. *Detection visualization from YOLOv11n showing parking slot occupancy classification (blue = car, cyan = free).*

As shown in Fig. 11, the model accurately identified and classified parking slots into two categories: ‘car’ (occupied) and ‘free’ (vacant). Bounding boxes were drawn with corresponding confidence scores, confirming consistent detection even under partial occlusions and variable camera perspectives.

The visualization results validate the model’s generalization performance, particularly in distinguishing tightly parked vehicles and empty spaces within complex urban parking lots. The confidence levels for detections remained high (typically above 0.8), indicating strong reliability for real-time applications. This visual verification further reinforces the quantitative metrics, demonstrating that YOLOv11n effectively handles spatial constraints, varying illumination, and multiple vehicle orientations across frames.

C. MongoDB Log Analysis

The integration of the detection outputs with a MongoDB real-time logging and analytics backend was done. Every slot detection is represented as a document with attributes slot IDs, occupancy (occupied/free), and time-stamped detection as illustrated in Fig. 11.

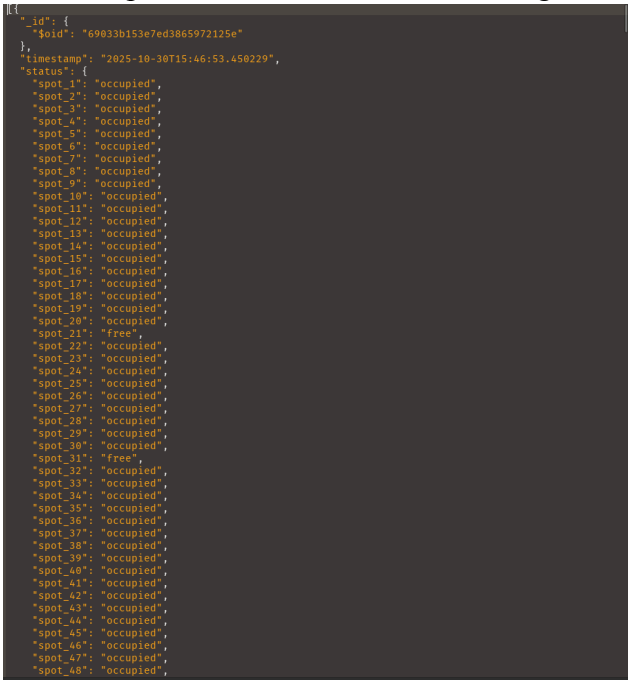

Fig. 11. *Sample MongoDB log document showing timestamped occupancy data for each parking slot. Each entry stores the slot ID, its current status (free or occupied), and a timestamp for temporal analysis and historical querying.*

Sample logs also attest to the fact that slot data and timestamps were recorded properly. The flexibility of the schema used in MongoDB can be utilized to store time-series properly and index historical analytics. Such logs allow the use of parking management authorities with post-processing and visualization at a scale without relational schema overheads.

D. System Output

The front-end dashboard displays the visualization of real-time parking slot availability integrated with the front-end as indicated in Fig. 12. All slots are color coded i.e. green one slot indicates free and red one occupied so that it can be simply monitored by the end user.

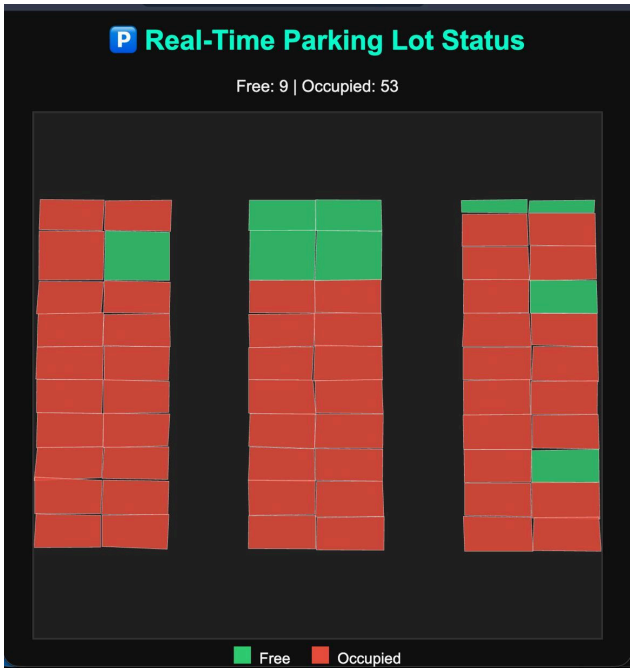

Fig. 12. *Real-time dashboard interface showing color-coded parking slot availability (green = free, red = occupied).*

The YOLOv11n inference is served on a Flask API and written to the MongoDB database at runtime and displayed on the dashboard about every 2 seconds. In the testing case, the interface reflected Free = 9 slots and Occupied = 53 slots which were correct and consistent with the backend and front-end responsiveness.

This visualization shows how the proposed system will be end-to-end, with detection, storage, and end-user analytics integrated into a smooth smart parking system.

V. Conclusion

The study introduces a fully-integrated YOLOv11n-based Smart Parking System, which is a combination of deep learning-based visual analytics with non-relational, scalable parking management through MongoDB storage. The framework proposed meets the end-to-end functionality, i.e., object detection and data persistence generated, to front-end visualization to provide actionable insights to the user in near real time.

YOLOv11n showed high levels of detection in terms of the mAP of 0.905, mAP-5095 of 0.613 and precision of 0.93 in a wide range of light and other environmental conditions. The findings confirm the system robustness and generalization as evidenced by constant loss convergence, and stable F1-performance across classes. It can offer time-series queries, trend predictions, and occupancy pattern analysis without the inflexibility of relational schemas by the backend integration with MongoDB to provide structured but flexible storage of slot-wise detections with timestamps.

The web-based dashboard also increases the usability of the system by displaying the occupancy status through the red-green slot indicators that are easy to interpret and periodically updated by the detection pipeline to maintain synchronization between the inference and display to the users. The existing implementation supports simulated video feeds as hardware access is limited, but the architecture is entirely based on real CCTV/RTSP input, and can be extended to support multi-camera deployments in urban areas.

The system will be further developed in the future with the objective of increasing the real-time abilities and predictive intelligence of the system. The initial step is to incorporate live RTSP/IP camera streams so that it will be possible to conduct the constant video

surveillance and make instant on-site inference. The framework will also use MQTT or WebSocket protocols to achieve low-latency communication between the detection module and the user interface in order to provide real-time updates to the dashboard without polling delays. On the data side, MongoDB aggregation pipelines will be used to automatically provide analysis results in the form of occupancy heatmaps, frequency of use reports, and anomaly notifications. In addition, optimization of the YOLOv11n model to run at a high speed with the help of TensorRT or ONNX will be considered to obtain high-speed inference with the lightweight embedded systems, such as Jetson Nano or Raspberry Pi. Lastly, future research will aim at predictive modeling of parking trends based on the study of the time series, so the system will be able to predict the high occupancy rates and suggest to the user the best parking zones to use in advance.

In general, this research forms a basis of an AI-based, database-optimized smart parking system bridging computer vision, big data and intelligent transport infrastructure - creating an opportunity for scalable smart city implementations.

References

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, “You Only Look Once: Unified, Real-Time Object Detection,” Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 779-788, 2016.

[2] L. da Luz, R. Costa, and P. Fernandes, “Smart Parking with Pixel-Wise ROI Selection for Vehicle Detection Using YOLOv8–YOLOv11,” arXiv preprint, arXiv:2412.01983, 2024.

[3] W. Jiang, Z. Liu, and M. Chen, “Two-Stage Efficient Parking Space Detection Method,” Applied Sciences (MDPI), vol. 15, no. 3, pp. 1123-1136, 2025.

[4] E. I. Vlahogianni, M. G. Karlaftis, and J. C. Golias, “Real-Time Parking Availability Prediction in Urban Mobility Systems,” Transportation Research Part C, vol. 142, pp. 104-118, 2022.

[5] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, “Benchmarking Cloud Serving Systems with YCSB,” Proc. ACM Symposium on Cloud Computing (SoCC), pp. 143-154, 2010.

[6] V. Abramova and J. Bernardino, “Benchmarking NoSQL Databases: Cassandra vs. MongoDB vs. HBase,” Proc. TPCTC Workshop on Performance Evaluation and Benchmarking, pp. 24-37, 2014.

[7] A. Davoudian, L. Chen, and M. Liu, “A Survey on NoSQL Data Stores,” ACM Computing Surveys, vol. 51, no. 2, pp. 1-43. 2018.

[8] N. Rizvi, S. Yadav, and A. Patel, “IoT-Based Smart Parking System for Urban Environments,” IEEE Access, vol. 11, pp. 81234-81245, 2023.

[9] M. Ahmed, F. Rahman, and T. Zaki, “Real-Time Smart Parking System Using YOLO11 and OpenCV,” ResearchGate Preprint, 2025.

[10] K. Yuldashev, A. Lee, and H. Kim, “Parking Lot Occupancy Detection with Improved MobileNetV3,” Sensors (MDPI), vol. 23, no. 4, pp. 1671-1682, 2023.

[11] J. Grbić and S. Koch, “Automatic Vision-Based Parking Slot Detection and Occupancy Classification,” Expert Systems with Applications, vol. 230, pp. 120-139, 2023.

[12] T. Nguyen and K. Sartipi, “Smart Camera Parking System with Auto Parking Spot Detection,” Proc. Asian Conference on Computer Vision Workshops (ACCV Wkshps), pp. 221-230, 2024.

[13] I. Martynova, L. Smirnov, and R. Karpov, “Revising Deep Learning Methods in Parking Lot Occupancy Detection,” arXiv preprint arXiv:2306.04288, 2023.

[14] X. Ma, “Review of Research on Vision-Based Parking Space Detection

Methods," IGI Global Journal of Intelligent Systems, pp. 45-61, 2022.