ParkLot: Centralized Parking Solution Using Fine-tuned Yolo Models

Department of Computer Science and Technology

National Institute of Technology Hamirpur

CONTRIBUTORS

Anirudh Sharma (22DCS002)

Pushpdeep (22DCS018)

MENTOR

Dr. Naveen Chauhan

Table of Content

Introduction & Motivation

Smart Cities Need Smart Parking

Efficient parking management is a critical component of building smart cities and improving urban mobility.

- **Stop the Search:** ***30%*** of urban traffic is caused by drivers looking for parking. Smart detection eliminates this loop.
- **Green Mobility:** Less **circling = less idling**. Directly lowers carbon footprint and fuel consumption.
- **Cost-Efficient Tech:** No new sensors needed. We unlock intelligence from existing CCTV networks.

Problem Statement

Fundamental Goal:

Achieve highly effective parking space detection

Domain Challenge: The system must work on aerial datasets (large area, top-down view), which are crucial for monitoring massive parking lots but are susceptible to issues like occlusions (e.g., one car blocking the view of another space)

Technical Question: Given the continuous evolution of real-time object detection models, which version of the YOLO family (v5, v7, v8, v9) offers the best performance.

Cost-Efficient Tech: No new sensors needed. We unlock intelligence from existing CCTV networks.

Literature Review

Ref & Year	Model/Approach	Main Work/Findings	Advantages	Limitations/Gaps
Amato et al., 2017	Deep Convolutional Neural Network (mAlexNet)	CNRPark-EXT, PKLot, custom	Lightweight decentralized detection on edge devices (Raspberry Pi)	+ Works on low-power devices; – Only binary (occupied/vacant)
Rafique et al., 2023	YOLO v5 (Object Detection)	PKLot, CNRPark-EXT	99.5% accuracy, real-time (~45 FPS)	+ High accuracy; ALPR-ready — needs more compute
Ahad & Kidwai, 2025	YOLOv4 + Behavioral Data	PKLot, CNRPark-EXT	Combines vehicle detection + user behavior → reduces parking search time	+ Better recall in crowded areas — Needs behavior sensors/data
Shreeram et al., 2025	YOLO Family (v5, v7, v8, v9)	Aerial datasets (e.g., EAGLE)	YOLOv9 highest accuracy (mAP≈0.994); YOLOv5 fastest	+ Good for large-area drone/CCTV — Aerial occlusions affect accuracy

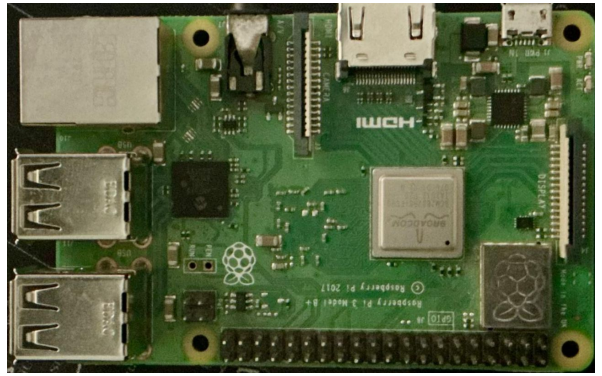
Datasets

Feature	CNRPark-Ext	PKlot
Primary Focus	Occupancy under difficult conditions (shadows, occlusion)	Large-scale detection across multiple lots & weather
Dataset Size	~150,000 labeled patches	~695,900 labeled patches
Scope	164 spaces (1 lot, 9 cameras, side view)	100+ spaces (3 lots, elevated views)
Conditions	Sunny, Overcast, Rainy (Pisa, Italy)	Sunny, Cloudy, Rainy (Curitiba, Brazil)

- Both these datasets provide a massive training pool of approximately **845,900 labeled slots.**.
- We also used **Custom Images** from Different Locations in **NITH** Campus for Accuracy Analysis and Testing.

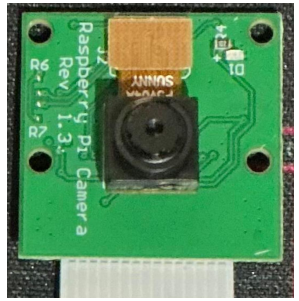
Hardware Specifications

- **Raspberry pi 3b+ ~ 5V-2.5A:**
  - **SoC**: Broadcom BCM2837B0, Cortex-A53 (64-bit) @ 1.4GHz
  - **Memory** (RAM): 1GB LPDDR2 SDRAM
  - **Storage**: 8GB (configured via MicroSD)
  - **OS**: Raspberry Pi OS Lite.



- **Logitech 720p webcam:**
  - **Connectivity**: usb webcam
  - **Working**: compatible with pi via usb2.0

- **Pi Camera:**
  - **Connectivity**: via raspi ribbon pin
  - **Status**: damaged (sensor not detected anymore)



- **GPU:**
  - P100 (kaggle), RTX 4060, 3050 (laptop)
  - for model - training

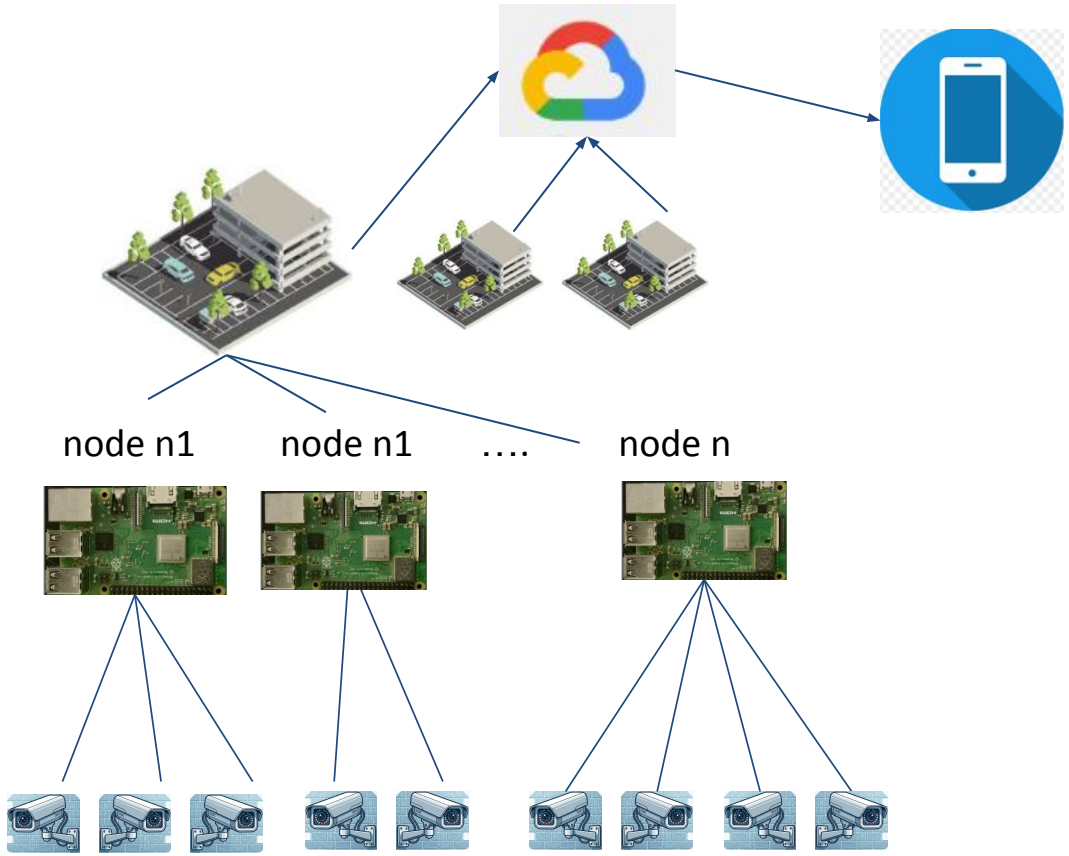
Cloud Server Specifications

- **VM Instance: (Google Cloud)**
  - **Processor:** Intel Emerald Rapids (c4-standard-2) with 2 vCPUs
  - **Storage:** 50 GB Hyperdisk Balanced
  - **RAM:** 7 GB
  - **IP Access:** Public IP *34.42.200.32*
  - **OS:** Debian 12 (Bookworm)

- **Storage: (GCS)**
  - **use:** Keep images for dataset collection.
  - unlimited storage with little cost per Gb (covered in credits)

- **Software or Frameworks:**
  - **Python Flask** server For Backend
  - **Docker** to deploy Flask Server on GCP
  - **React.js** for Frontend
  - **MongoDb** database

Our Solution



Yolo Detection Model

Parklot leverages computer vision to detect vehicle presence using standard CCTV feeds.

- **Real-time Detection:** Uses Object Detection to identify a slot as empty or occupied based on overlapping annotations
- **Edge Processing Models:** Uses edge processing to reduce network bandwidth for image data transfers.
- **User App:** Directs drivers to the nearest available spot via Our app based on parking location.

  Dashboard Monitoring - Multiple camera Nodes (parking spaces) can connect to a single parking Lot location.

Methodology

1. Model Fine-tuning
2. Yolo model comparison and Results
3. Cloud Server Development
4. Edge Device Integration
5. Frontend Integration

Model Training (Dataset Processing)

Goal: Convert Occupancy Dataset → Object Detection Dataset

- **Phase 1: Structure Initialization**
  - Create parallel directory structure for train, val, test.
  - Establish writable paths (handling read-only input sources)

- **Phase 2: Resource Optimization**
  - Symlinking: Create symbolic links for 140k images instead of copying.
  - Reduces storage overhead to near zero while maintaining access

- **Phase 3: Label Normalization (Core Logic)**
  - Parse original annotations (Class 0: Free, Class 1: Occupied).
  - Merge: Force all class IDs to 0 (Car).
  - Preserve original bounding box coordinates exactly.

- **Phase 4: Configuration Deployment**
  - Generate data.yaml defining the single class ('car').
  - Verify integrity of new dataset structure.

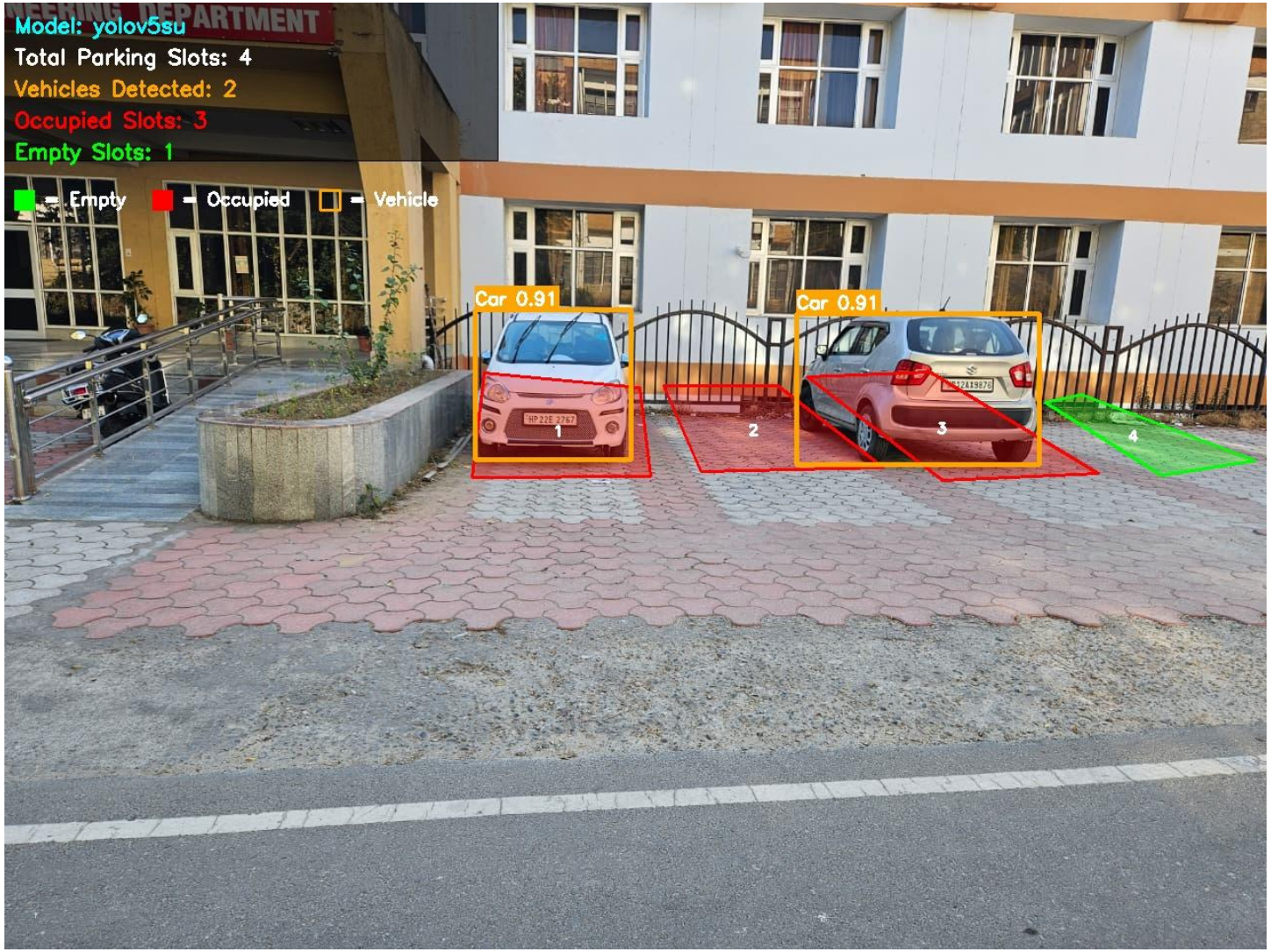
Model Training (Training Yolo Models)

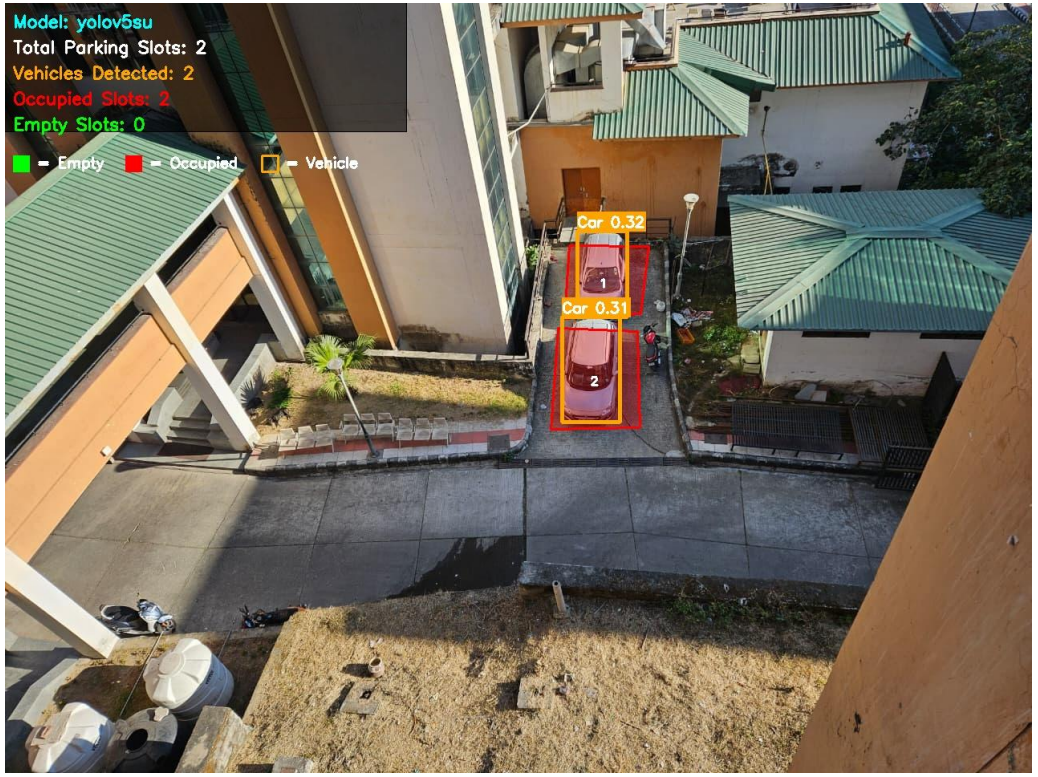
- We have trained and tested multiple yolo models for object detection like **yolov8m, yolov8l** and nano models like **yolov8n** for edge devices.
- We have used mainly 2 datasets for training the larger models- **CNRpark-Ext** and **Pklot**.
- We also used a **custom dataset** from our campus parking lot images, which was annotated using **CVAT.ai** **tool** and **labels** were downloaded for training on yolov8n.
- We Compared accuracy of the output produced using Precision, Recall and F1 Score, as well as visually checking the image output.

Model Training (Results Analysis)

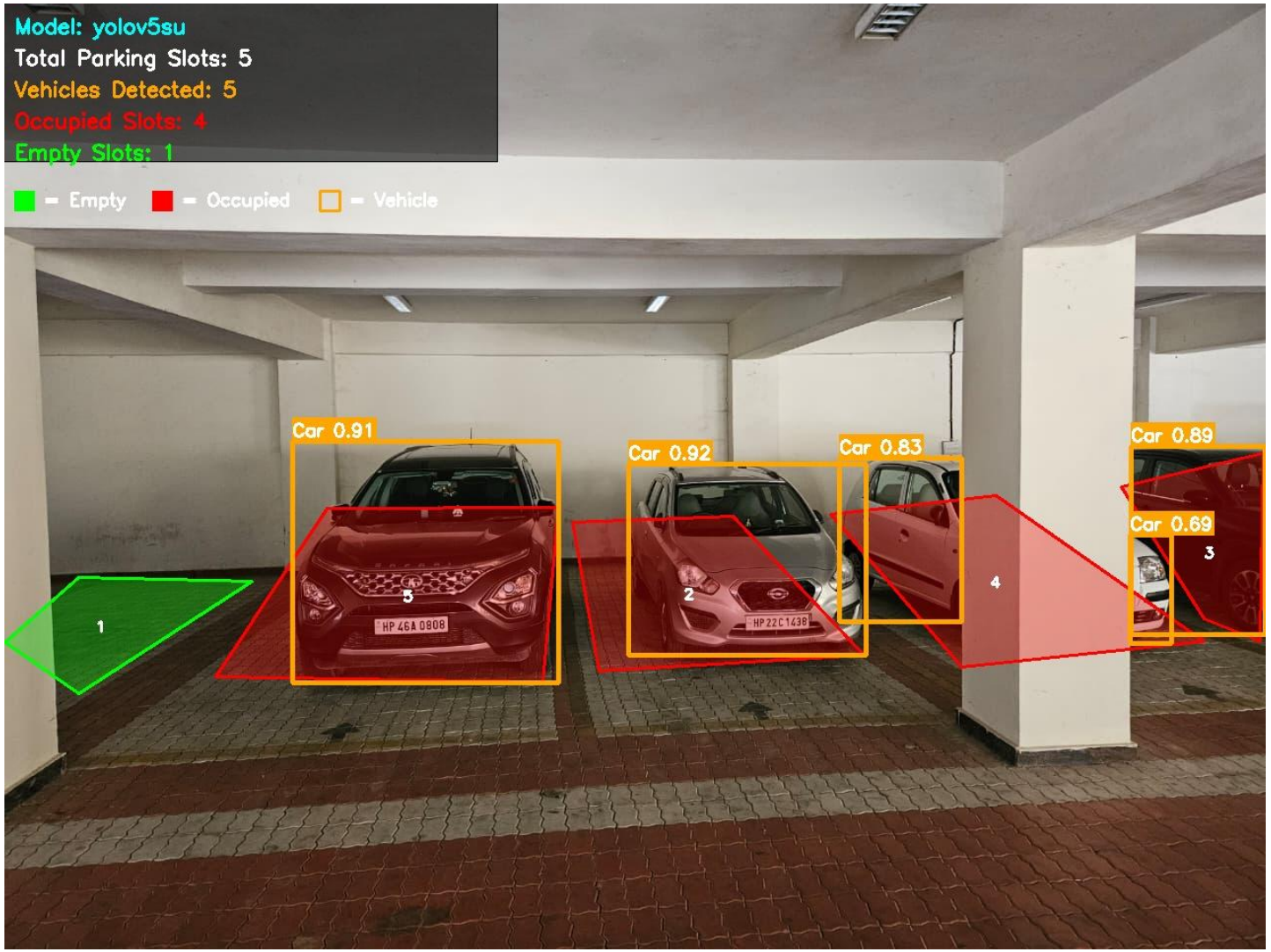
Model Name	Precision	Recall	Inference Time (ms)	FPS	F1-Score	Total Vehicles	Vehicles per Image
YOLOv8m-cnr-subset	0.92	0.935	257	3.89	0.927	366	18.3
YOLOv8m-parking-subset	0.9	0.92	280.03	3.57	0.91	374	18.7
YOLOv8l-cnr-subset	0.935	0.95	387.69	2.58	0.942	341	17.05
YOLOv8l-parking-subset	0.915	0.93	486.04	2.06	0.922	354	17.7
YOLOv8n-custom (Pi)	0.99	0.882	7.7	100	0.933	108	4
YOLOv8n-cnr-subset	0.865	0.895	56.79	50	0.88	384	19.35

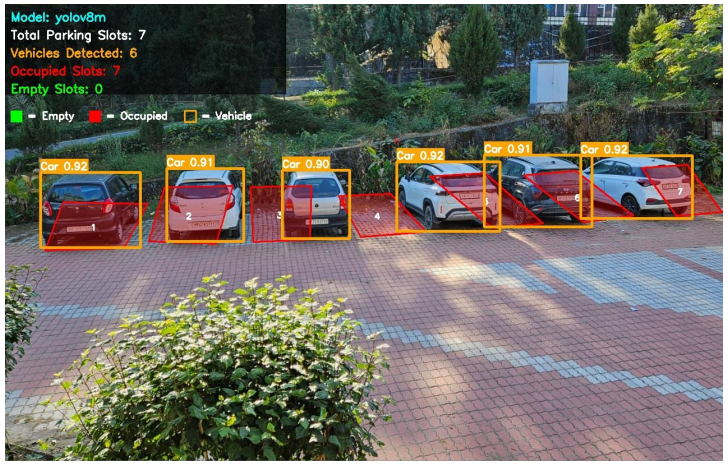
Results Analysis - Campus Images (Cont.)

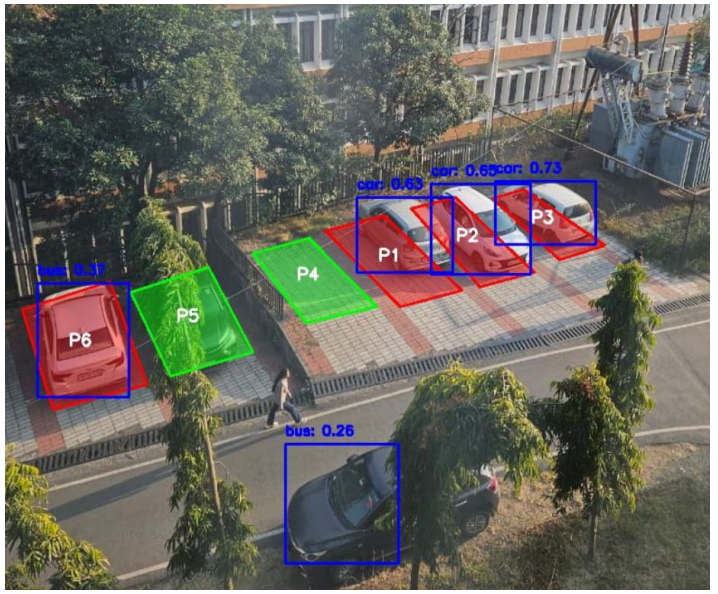




Results Analysis - Campus Images (Cont.)





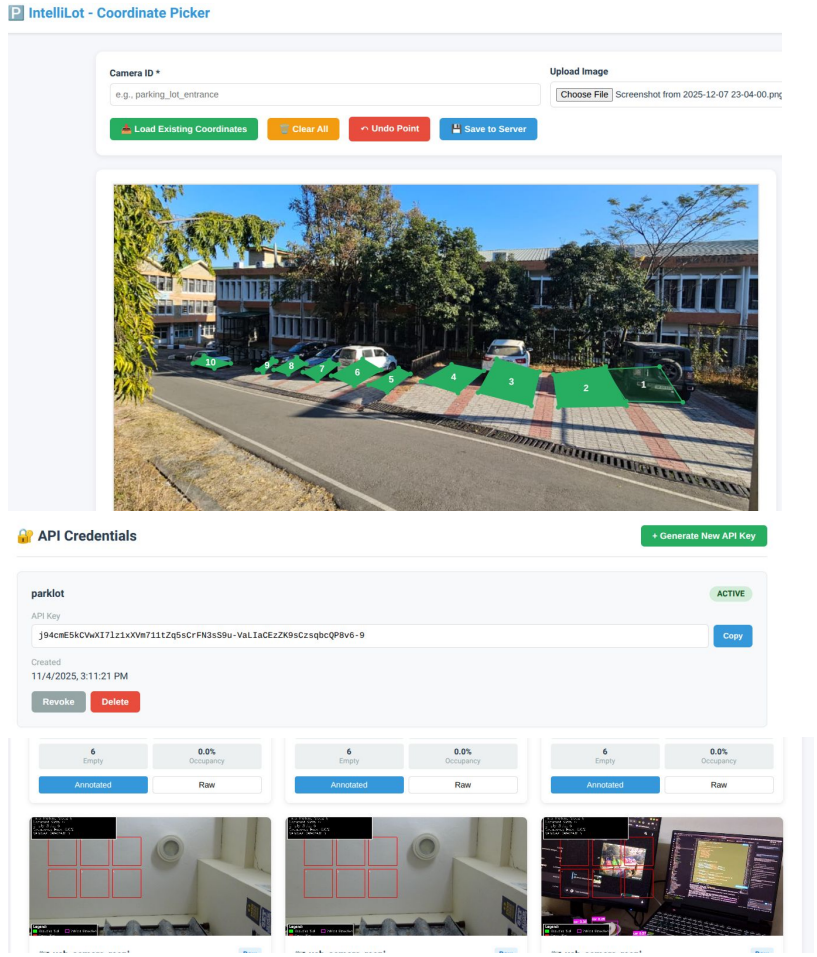


Cloud Server Development (http://34.42.200.32)

- **Organization Dashboard:**
  - Create Api for Parking Space
  - Multi-Nodes and Cameras Management
  - Faster Annotations using visuals

- **Database & TechStack:**
  - Built on Flask, deployed using docker, Uses MongoDb Database
  - Provide api's to store results to database and update parking status of the particular parking node/camera

- **API Access:**
  - **/udpate:** update the database using edge processed data
  - **/udpdateRaw:** process raw image using our model and store raw and annotated image to gcp and update status
  - **/getdata:** get the status of parking lot
  - **/getimages:** get the saved images from gcp for monitoring
  - **/auth:** to authenticate the organization user to dashboard

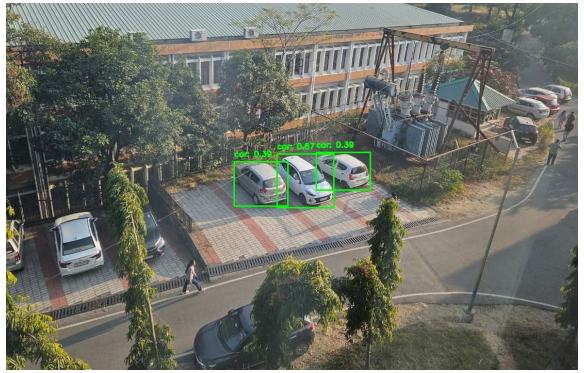


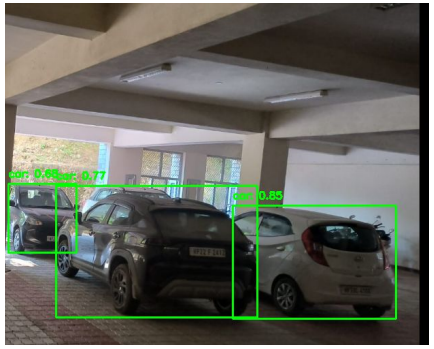
Edge Device Testing & Integration

- We are running a **custom python server** script on the raspberry pi along with a **usb webcam** to capture image.
- The connection to server is established via an **api key** which is generated on the user dashboard through our **cloud server**
- **The config for the pi looks like:**

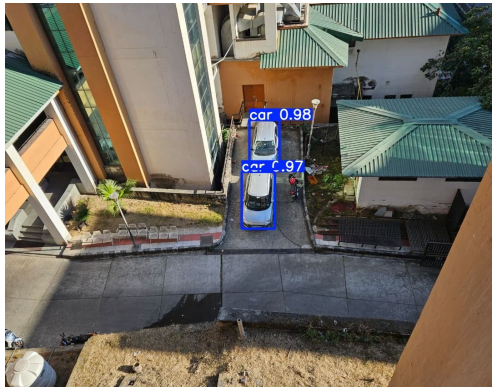

```
{
    "camera_type": "web_upload",
    "api_endpoint": "http://34.42.200.32/parking/updateRaw",
    "api_key": "j94cmE5kCVwXI7lz1xXVm711tZq5sCrFN3sS9u-VaLIaCEzZK9sCzsqbcQP8v6-9",
    "camera_id": "web_upload_camera",
    "node_id": "web_server_node",
    "interval": 60,
    "device_id": "web_server_node",
    "retry_attempts": 3,
    "retry_delay": 5,
    "save_local_copy": true,
    "local_save_path": "./uploads"
}
```

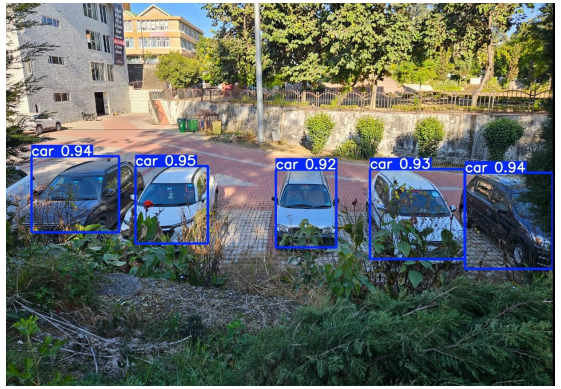
Edge Device Testing & Integration (cont.)





a) yolov8n detection on raspberry



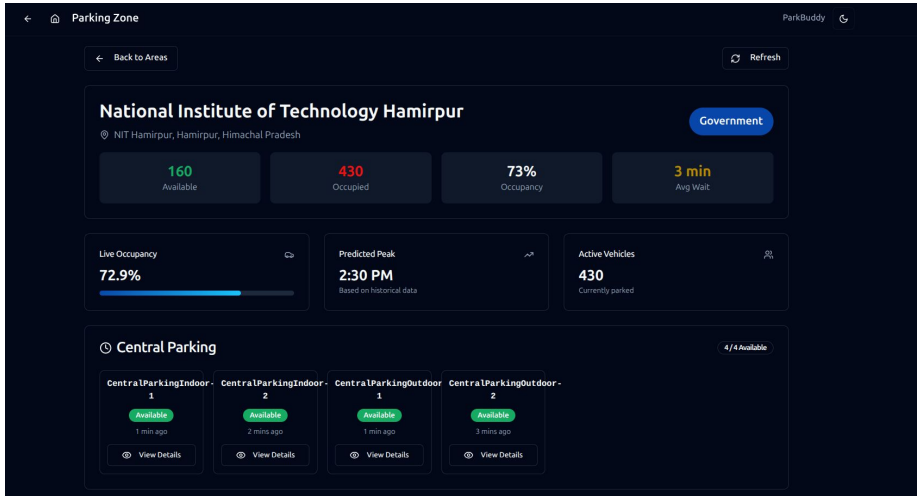


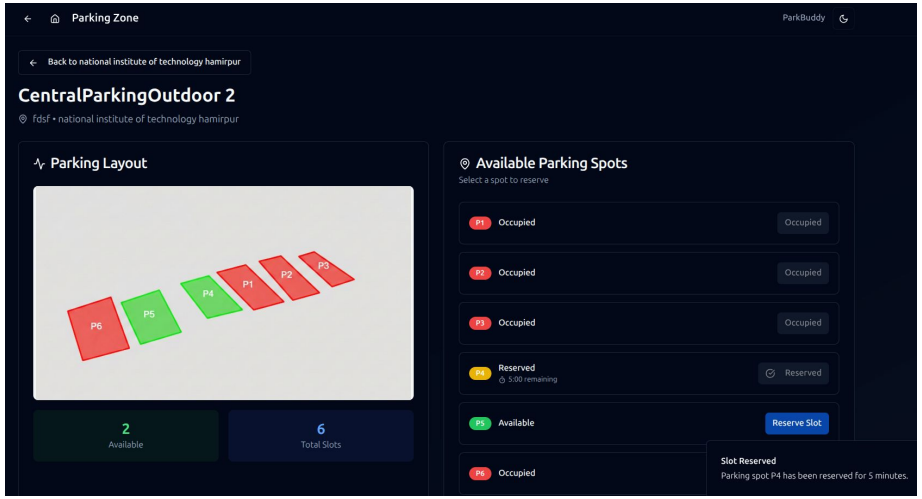
b) yolov8n finetuned on nith images

Frontend App Integration

- **Tech Stack:**
  - React.js
  - Tailwind.css
  - Backend Api's using javascript

- **Features:**
  - Database Access
  - Real Time Availability Check
  - Multiple Parking Space at One Parking Lot
  - Parking Image Data Abstraction
  - Parking Reservation System (time based)





Conclusions

- **Mitigates Urban Congestion:** Directly addresses the 30% of city traffic caused by drivers searching for parking, significantly smoothing traffic flow.
- **Cost-Efficient Retrofit:** Removes the need for expensive sensors by unlocking intelligence from existing CCTV camera networks.
- **Optimized Hybrid Architecture:** Balances edge computing on Raspberry Pi nodes with cloud scalability to ensure real-time performance without network lag.
- **Promotes Green Mobility:** Reduces carbon footprint and fuel wastage by minimizing the "circling" and idling time required to find a spot.
- **Robust Full-Stack Implementation:** Successfully integrates YOLO detection models with a Flask and MongoDB backend for reliable data storage and API access.
- **Seamless User Experience:** Connects drivers to available spots instantly through a mobile app, bridging the gap between detection and navigation.

References

- [1] G. Amato, F. Carrara, F. Falchi, C. Gennaro, and C. Vairo, "Deep learning for decentralized parking lot occupancy detection," *Expert Systems with Applications*, vol. 72, pp. 327-334, Apr. 2017
  Source: **https://doi.org/10.1016/j.eswa.2016.10.055**

- [2] M. A. Rafique, A. Gul, S. Jan, and F. Khan, "Optimized real-time parking management framework using deep learning," *Expert Systems with Applications*, vol. 220, p. 119686, Jun. 2023.
  Source: **https://doi.org/10.1016/j.eswa.2023.119686**

- [3] A. Ahad and F. A. Kidwai, "YOLO based approach for real-time parking detection and dynamic allocation: integrating behavioral data for urban congested cities," *Innovative Infrastructure Solutions*, vol. 10, no. 6, 2025.
  Source: https://www.researchgate.net/publication/392128836_YOLO_based_approach_for_real-time_parking_detection_and_dynamic_allocation_integrating_behavioral_data_for_urban_congested_cities

- [4] S. Hudda, et al., "Smart Parking Space Availability Detection Using Aerial Images and YOLO-Based Deep Learning Models," in Proc. of the 39th International Conference on Advanced Information Networking and Applications (AINA-2025), 2025.
  Source: https://ieeexplore.ieee.org/document/11187502

- Link to code: https://github.com/orgs/IntelliLot/repositories

Q & A

Thank you for your attention.

Anirudh Sharma - 22dcs002

Pushpdeep - 22dcs018