Chaminda Kodithuwakku

VEHICLE REGISTRATION-PLATE DETECTION WITH MACHINE LEARNING

A Practical Approach

Master Of Cloud Based Software Engineering
2024

ACKNOWLEDGEMENT

I would like to express my heartiest gratitude to my supervisor and our principal lecturer, Johan Dams. His continuous support, insightful feedback and invaluable guidance support me to deliver this thesis successfully.

I am also grateful to my previous working place. They allowed me to access their resources and gave me invaluable advice related to my thesis.

A special thanks to my wife, kids and friends for their encouragement at any time whenever I called.

Finally, I appreciate all the researchers, contributors in the Machine Learning and computer vision whose work inspired me and guided me to do this research.

Thank you all for making this journey fruitful and filled with rewarding experiences.

Chaminda Kumara Kodithuwakku

25th March 2025

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Cloud-Based Software Engineering

ABSTRACT

Author	Chaminda Kodithuwakku
Title	Vehicle Registration-Plate Detection With ML A Practical Approach
Year	2024
Language	English
Pages	73 + 6 Appendices
Name of Supervisor	Johan Dams

The thesis about vehicle registration plate detection, which is based on the problem of the existing parking system, PERCS. The parking system uses a third-party product (VRS) for vehicle recognition. But PERCS stakeholders decided to replace the VRS system and introduce their own mechanism to recognize vehicles. As a result, surveillance cameras (as IoT) were introduced for parking areas to capture vehicles and send them to another API service to identify vehicle's details. In that case, all the captured images are sent to a vehicle recognition API. Sometimes, license plates of many vehicles are not visible in the images. Therefore, the vehicle recognition API returns empty result when it sends vehicles without license plates. PERCS management wanted to ignore vehicles without license plates for image recognition API, as it incurred additional costs at the end of month.

To address this issue, we decided to identify licence plate availability in the image before it sends to the vehicle recognition API. The task started with finding a suitable machine learning model. Also, required images to train the model are taken from PERCS system. This implementation hosts in MS Azure environment because PERCS systems are mostly hosted in Azure Web Apps.

Keywords License Plate Detection, Cloud-based Parking Systems, ML, IoT

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ACI – Azure Container Instance

AKS – Azure Kubernetes Services

ANPR – Automatic Number Plate Recognition

API – Application Programming Interface

CNN – Convolutional Neural Network

CPU – Central Processing Unit

CV – Computer Vision

DevOps – Development and Operations

DVC – Data Version Control

Fast R-CNN – Fast Region-Based Convolution Neural Network

Faster R-CNN – Faster Region-Based Convolution Neural Network

GDPR - General Data Protection Regulation

GPU – Graphics Processing Unit

HIPPA - Health Insurance Portability and Accountability Act

HOG - Histogram of Oriented Gradients

HTTP – Hyper Text Transfer Protocol

I/O – Input/Output

IEC - International Electrotechnical Commission

IoT – Internet of Things

IR – Infrared Radiation

ISO - International Organization for Standardization

LPR – License Plate Recognitions

ML – Machine Learning

MLFlow – Machine Learning Flow

MLOps - Machine Learning Operations

MQTT - Message Queuing Telemetry Transport

ONVIF - Open Network Video Interface Forum

PCI-DSS - Payment Card Industry Data Security Standard

QoS - Quality of Service

R-CNN - Region-Based Convolution Neural Network

RNN – Recurrent Neural Network

RTSP - Real-Time Streaming Protocol

Rx - Reactive extensions

SIFT - Scale-invariant Feature Transform

SSL - Secure Sockets Layer

SURF - Speeded-up Robust Features

SW – Software

TCP – Transmission Control Protocol

TLS - Transport Layer Security

vNet - Virtual Network

VRS – Vehicle Recognition System

WS – Web Socket

YOLO – You Only Look Once

1 INTRODUCTION

Vehicle detection with machine learning (ML) helps us to build applications like traffic monitoring, smart parking systems and autonomous driving (Neamah & Karim, 2023). Also, it is an important technology in modern urban planning (Mohanty et al., 2016). Vehicle numbers increase with expansion of cities, so it is essential to monitor and manage vehicle flow using intelligent systems. On the other hand, conventional vehicle detection techniques as human counting or simple sensors are inaccurate, ineffective, and frequently imprecise when dealing with complicated environments. ML, particularly with advances in deep learning and computer vision (CV) offers a solution that is both scalable and highly accurate.

Machine Learning algorithms are used in vehicle detection system to recognize vehicles in a scene, frequently in real time by analysing pictures or video streams. As leading object detection models, Convolutional Neural Networks (O'Shea & Nash, 2015), YOLO (Redmon et al., 2015) have been used effectively for this purpose. These models are useful tools for automated systems and urban surveillance (Oguine et al., 2022) since they can identify cars in a variety of situations, including those with partial obstacles, intense traffic, and light or dark conditions.

Vehicle detection powered by machine learning serves many purposes. It can be used as a smart parking system to reduce the time a driver spends searching for a free space, by identifying vehicles and directing them to available spots without human intervention. In autonomous vehicles (Garikapati & Shetiya, 2024), ensure safe navigation with real time vehicle recognition. Also, ML based vehicle detection systems allow security sectors to make decisions based on the data as real time count and flow analysis (Alqahtani & Kumar, 2023). Furthermore, vehicle detection with ML helps to build smart and connected cities (Mohanty et al., 2016) with improving safety, efficiency, and convenience. This thesis focuses on explore ML based techniques for vehicle detection, evaluate them and their effectiveness in real world scenarios.

2 PURPOSE, OBJECTIVES AND RESEARCH QUESTIONS/TASKS

2.1 Purpose

The Parking Enforcement Revenue Collection System (**PERCS**) (Omniq, 2024) was designed to optimize and automate the processes involved in parking enforcement and revenue collection in united states. It is often used by municipalities, parking authorities, and private organizations to manage parking spaces, enforce parking regulations, and collect associated revenues efficiently.

Currently, PERCS is using different kind of image recognition technologies to capture real time vehicle information. HTS (Bliccathemes, n.d.-a) is the main vehicles recognition systems (VRS) provider for PERCS.

PERCS is using HTS products such as N70 -IP camera for medium speed license plate recognition, LC3000/3100 Fan-Less lane controller, LC4000 Rackmount server, SeeControl Client SDK, Controller Application and SeeControl (Bliccathemes, n.d.-b). These products are installed on the car park traffic lanes entrances, exits and internal locations. The most basic setup of the system includes a single camera which is mounted on a pole at the side of the lane, front view and the rear view can be captured by camaras when a vehicle passing along the road.

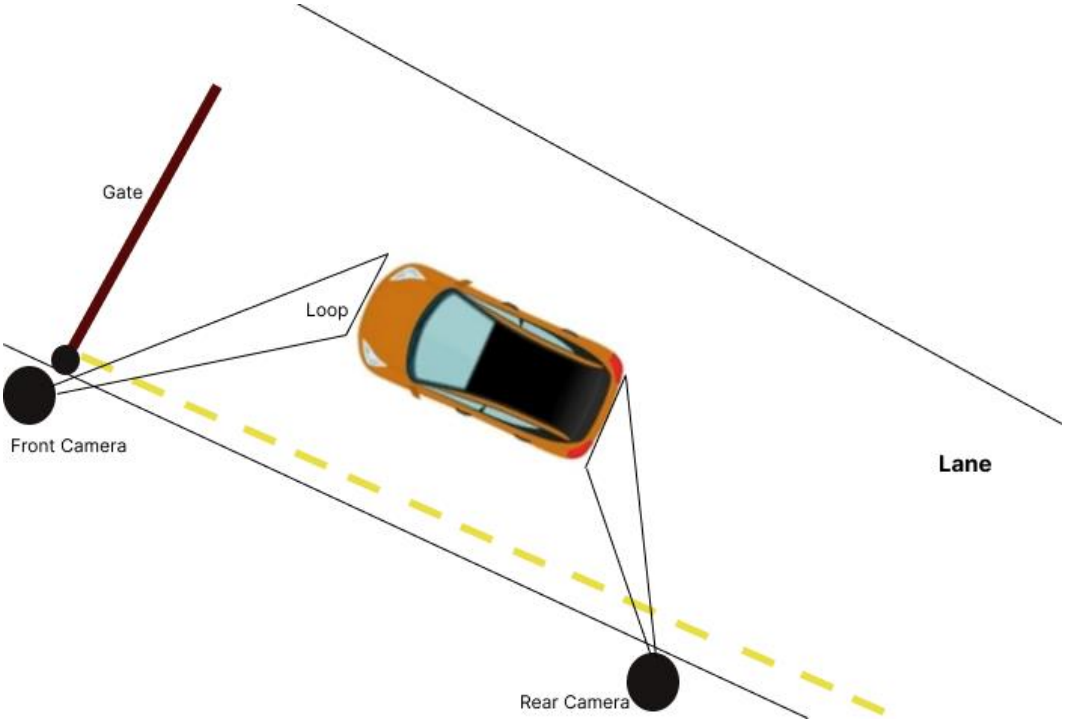


Figure 1. A front and rear capture LPR system.
Figure created by author, based on A Division of Quest Solution Inc (2019).

A lane controller (Bliccathemes, n.d.-c) is often placed inside a gate cabinet or an enclosure near to the lane. Each unit might be triggered by a loop detector or SW trigger. The camera takes many visible and/or near-infrared (IR) snapshots of the vehicle. After that, it notifies the central server with its result. The system uses License Plate Recognition (LPR, sometimes called ANPR) (HTS Vehicle Recognition Solutions, n.d.) to extract the plate number. Then, image files are transmitted over a network and stored on the disk. While the system can notify users of vehicles who are already in the blacklist or open the gate for permitted vehicles on a white-list.

Also, the recognition system is based on imaging units which capture and process front and rear images of vehicles. The imaging unit includes a powerful camara and builds in LED illumination units. N70 (HTS, n.d.) is the most usable imaging unit which is the best fit for parking systems. Lane controller (Bliccathemes, n.d.-d) manages the imaging units. This lane controller uses various triggers to determine when to take the images. It adjusts the camera's settings for optimal image

capture. The lane controller reads the captured images and sends them to well-known HTS recognition engine to extract the license plate and other parameters. Finally, it sends the extracted data to the VRS server (HTS Vehicle Recognition Solutions, 2021).

The VRS consists of a network of imaging units. These imaging units are connected to number of lane controllers. The lane controllers relay the collected data to one or more VRS servers (HTS Vehicle Recognition Solutions, 2021). The system has a modular architecture that is scalable and adaptable. This modular design helps to do multiple configurations and helps to address the unique needs of different projects.

After receiving the intermediate results from the VRS lane controllers, the VRS server (powered by the SeeControl service) calculates the results. It uses a multi-platform web application to display events (HTS Vehicle Recognition Solutions, n.d.-a). The events are stored in an MS SQL database and the system allows the user to monitor nine outcomes. Users can query past events, and they receive alerts based on pre-configured lists (Hotlist). It can export the data to external systems using an interface that is either customized for a particular interface type or supplied via an SDK.

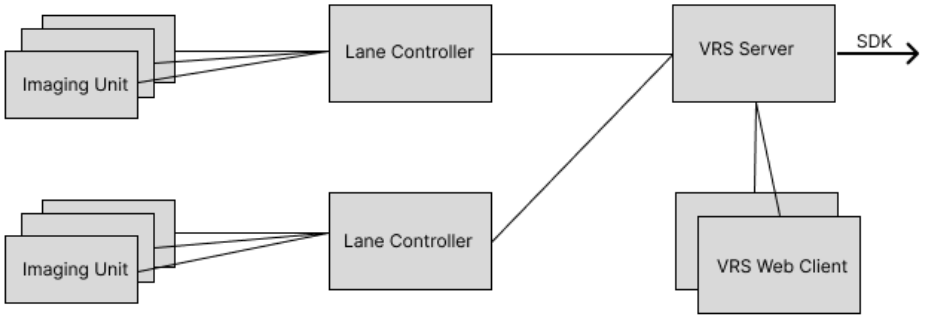


Figure 2. VRS Architecture
Figure created by author, based on HTS (2019).

PERCS has been integrated with VRS in many projects. But the maintenance and initial cost of each project is going to high day by day. So, the stakeholders of PERCS have decided to replace VRS with another solution for PERCS. **Cost reduction is the main purpose of VRS replacement.**

2.2 Goals

- Identify suitable vehicle detection camera: Capable camera with IoT + Edge computing features to detect vehicles, capture vehicles and send them to cloud service.
- Develop an efficient vehicle detection system: Capable to identify vehicles with license plates accurately in real time.
- Optimize ML Model: Fine tune for speed and accuracy in vehicle detection aiming minimal latency.
- Implement real world Scenario: PERCS is a leading parking system in US and planning to introduce an alternative implementation for the existing vehicle recognition system.

2.3 Research Questions/Tasks

The research will be discussed how to distinguish vehicles with visible license plates from other objects.

Our main goal is cost effective implementation in PERCS system to identify vehicles when they enter to the parking area. Then, only vehicles with license plates will be sent to the Image Recognition service.

- **Primary Question**

  How can machine learning be effectively used to achieve accurate and efficient license plate detection for real-time vehicles in diverse environments?

- **Supporting Questions**
  - What is the most suitable learning model for real-time vehicle's license plate detection accurately?
  - How to compare performance of different models in vehicle detection accuracy, latency, and the computation efficiency?
  - What kind of dataset (Public, custom, annotated) need to train for vehicle detection scenarios?
  - What software frameworks (e.g.: ONNX, TensorRT) and hardware (e.g.: GPUs, edge devices) are most effective to deploy the ML model in the real application?
  - How does data augmentation impact to the model for generalized unseen conditions?
  - What optimizations (model compression, hardware acceleration) to do for meet the requirements for real time detections?
  - How to manage challenging conditions like low light, misty, bad weather?

2.4 Outcomes

- Cost Deduction: Currently using an expensive product to capture and process vehicle details in the PERCS, which can be replaced after this implementation.
- Implementation Plan: A technical guide should be prepared for new implementation, including required devices, software architecture, and system integration steps.

3 LITERATURE REVIEW

This chapter provides a critical analysis of existing research, relevant theories, and findings related to image detection of vehicles. End of the analysis, it provides a comprehensive overview of what is known, identify gaps in the research and establish the context and justification.

3.1 Overview of Vehicle Detection.

Machine learning offers several types of techniques to detect vehicles. Therefore, Various factors should be considered to choose a suitable technique for the specific task. These factors are complexity, accuracy, and computational demand. The main objective of this survey is to identify the most suitable machine learning technique to use for vehicle detection.

Vehicle detection is linked to image recognition techniques, and it is a part of computer vision. In computer vision, the field of image recognition is responsible for identifying objects, features, or patterns in a digital image. Different models and techniques are used to process and interpret visual data. Also, Image recognition techniques are categorized into two groups, they are traditional computer vision methods and deep learning approaches. (Karypidis et al., 2022).

In traditional techniques, several feature extraction techniques have been used including Speeded-up Robust Features (SURF) (Bay et al., 2006), Oriented FAST, Scale-invariant Feature Transform (SIFT) (Bay et al., 2006), and Rotated BRIEF (ORB) (Wang et al., 2015). Also, Object detection has been performed using Histogram of Oriented Gradients (HOG) (Terayama et al., 2009), which focuses on the structural and shape features of objects. Likewise, various kinds of techniques based on mathematical models and algorithms have been used to process and extract features from images. These methods are not related with deep learning or any datasets.

3.2 Deep Learning in Vision Tasks.

Neural network (Schmidhuber, 2014) was introduced in 1943 by Warren McCulloch and Walter Pitts. They developed mathematical models of artificial neurons. This was the foundation of modern neural networks. In 1958, Frank Rosenblatt invented the Perceptron, the first trainable neural network which makes a remarkable step in artificial intelligence. However, it was unfortunately put on hold due to a lack of computing power and theoretical challenges. Again at 1980s, Geoffrey Hinton, Yann LeCun and the team energized neural network with backpropagation (Sekhar & Meghana, 2020). At 2010s, the deep learning era began with advance GPUs, large datasets, and improved algorithms (M. Li et al., 2024).

After Introducing Deep learning techniques, technical experts discovered that mixing the flavour of deep learning with computer vision gives a powerful outcome to identify attributes in objects (Chen et al., 2024). This literature presents different kind of deep learning algorithms that are effectively used in Image recognition.

Neural networks are divided into different types, and they have developed for some specific use cases. For instance, recurrent neural networks are widely used in natural language processing tasks (Yin et al., 2017). On the other hand, convolutional neural networks (CNN) (Premaratne et al., 2023) are mostly used for classification and computer vision tasks. Furthermore, leading techniques like YOLO (You Only Look Once) (Redmon et al., 2015), region-based CNN (R-CNN) (S, 2020), Fast R-CNN (Girshick & Microsoft Research, n.d.), Faster R-CNN (Ren et al., n.d.) will be discussed and comparison to identify the best approach to vehicle detection.

Specially, R-CNN has a key concept called region proposals (S, 2020). The region proposals help to identify each part of the image as existing objects of the image. Also, by allowing the model to focus on smaller regions that are most likely to contain objects, they improve both accuracy and efficiency.

R-CNN models define regions which contain objects. These region proposals pass through a CNN and extract the features, after that they will classify to determine the class and refine the bounding box. This process continues in each region independently and consumes more computation power. Therefore, R-CNN works really slow in real time applications.

Although, Fast R-CNN (Girshick & Microsoft Research, n.d.) is the enhanced version of R-CNN. It reduces computing time by using the entire image at once passing through a CNN, it does not consider each region separately. But regional proposals are still used in Fast R-CNN as a shared feature map. Shared feature map speeds up the process than R-CNN (Girshick et al., n.d.)

The Faster R-CNN model introduced as an improvement over Fast R-CNN (Ren et al., n.d.). It includes an integrated region proposal network instead of an external region proposal algorithm as used in Fast R-CNN. This integration is the major reason to superior performance compared to predecessor of Faster R-CNN. Currently, Faster R-CNN remains a widely used model for various object detection tasks (Ren et al., 2015).

YOLO (You Only Look Once) (Boesch, 2025) is another leading real time object detection model with high accuracy. Applications like self-driving cars (Azevedo & Santos, 2022), video analysis (Jana et al., 2018), and robots require almost instantaneous object detection (Reis et al., 2019). These applications can now analyse data quickly and efficiently because of YOLO's revolutionary redefining object detection models. Key characteristics of YOLO like Single pass detection, grid-based detection, bounding boxes and class probabilities make it a high speed and efficient model.

Also, YOLO has evolved, and it has improved from its initial release amazingly. Object detection is simplified in YOLO by treating it as a simple task. So, the entire image processes at once and passes through the network to outputs as bounding boxes and class probabilities concurrently. According to this revolutionary change,

YOLO works fast and perfectly matches the real time application. Also, Single pass detection performs better than region-based implementation of R-CNN.

Further, YOLO defines a grid for input image and divided into cells that equally rows and cells (5x5, 10x10, 11x11, etc…). Also, every grid cell has a responsibility to predict the presence of an object, because the object may become the centre cell of the presence. With this configuration, YOLO may predict several items in various regions of the image without requiring separate region proposals (Redmon et al., 2015).

Moreover, YOLO predicts bounding boxes and corresponding confidence values for every grid cell. It also determines the kind of object in each bounding box by predicting class probabilities for each box (Redmon et al., 2015). Again, YOLO keeps the bounding boxes with high confidence scores and rest of low confidence boxes to reduce false positives. And it is super-fast because YOLO reads the image once, it doesn't perform more passes like R-CNN.

However, with the invention of Transformers, Transformer model-based vision transformers (Dosovitskiy et al., 2020) and detection transformers (Carion et al., 2020) have been introduced recently. Basically, transformers have two components, an encoder and a decoder. Each component has layered architecture, and they contain fully connected neural networks. Transformers can parallelly process the entire sequence at once, faster than RNNs. Other than that, transformers are fully capable of keeping the long-range dependencies among the tokens. But high computational and memory cost are the main concerns of the Large Multi models.

3.3 Model comparison for Object Detection

There are many deep learning models available in the market. Some of them are highly accurate and others respond quickly. YOLOv8 (YOLOV8: State-of-the-Art Computer Vision Model, n.d.), SSD (Liu et al., 2016), RetinaNet (T. Lin et al., 2017), Faster R-CNN (Ren et al., 2015), Mask R-CNN (He et al., 2018) and DETR (Carion et

al., 2020) are some of the mostly used models in the industry for object detection purposes. In this case, determining the most suitable model for PERCS among these models (Sojasingarayar, 2024).

Table 3-1. Models Comparison
Adapted from Hui (2018), Boesch (2025), Carion et al. (2020a) and He et al. (2018).

Model	**Speed (FPS)**	**Accuracy (mAP@50) %**
Yolov8	Fast (~280)	73-85
RetinaNet	Medium (10-20)	38-39
SSD	Fast (60 -100)	41-58
Faster R-CNN	Slow (1-10)	41
Mask R-CNN	Slow (5-16)	77
DETR	Fast (42 -90)	39-50

Accuracy, speed and strongness in different circumstances are the major factors of the best model for license plate detection. According to the above Table 3-1, YOLOv8 provides real-time performance, it is highly efficient with strong accuracy. Also, YOLOv8 is ideal for fast inference applications like traffic monitoring. On the other hand, SSD (Single Shot MultiBox Detector) is another fast model, but it is difficult to identify small objects like license plates. Also, Retina Net, which is slower than YOLO. But it performs better than SSD to detect small objects. Faster R-CNN is a high accuracy model since its region proposal network. But it is computationally more expensive than YOLO, SSD and Retina Net. Comparing YOLO, SSD, Faster R-CNN, and RetinaNet, Faster R-CNN can be considered as a weak model for real-time applications due to its slower inference speed. However, Mask R-CNN is an extension of Faster R-CNN. It has been developed to handle both object detection and instance segmentation tasks more effectively. DETR (Detection Transformer) which is based on Transformer, and it has a massive capability to detect objects in complex scenes without using anchor boxes, but it needs a large dataset, and it is slower than YOLO.

Based on this comparison, YOLOv8 is the most suitable choice for object detection. But YOLO is not a perfectly accurate model for small objects like license plates. Accuracy is the highest priority for parking systems. According to that, Faster R-CNN or RetinaNet can be considered as a better choice.

Table 3-2. Faster R-CNN vs RetinaNet

Model	**Speed (FPS)**	**Accuracy (mAP)%**	**Complexity**	**Real-Time Detection**	**Performance on Small Objects**	**Deployment**
RetinaNet	Fast	High for Small Objects Moderate	Moderate	Good with small plates	Effective for small plates	Suitable for low-power, embedded systems.
Faster R-CNN	Slow	Especially Very High for small, obstructed objects	More Complex	Not Ideal for Real-Time	Very Effective for small Plates	Suitable for high-end systems with GPU.

3.3.1 Faster R-CNN vs RetinaNET

Both RetinaNet and Faster R-CNN provide strong object detection capabilities, but they are in different computation efficiency (speed, accuracy) (Faster RCNN Vs Retinanet Comparison | Restackio, n.d.).

Faster R-CNN can be accepted as a high accurate model since its two-staged detection pipeline, it starts with identifying Regional Proposal Network (RPN) and ensuring that potential objects before classification. This is mostly suitable for purposes that requires high accuracy like automated toll systems or law enforcement applications. However, considering its slowness (~1-10 FPS), it cannot recommend for real-time cloud inference without powerful GPUs (Weihong & Jiaoyang, 2020).

Also, RetinaNet is a single stage object detector. It has well balanced speed and accuracy. While its mAP also slightly lower, it uses focal loss (Lin et al., 2017) to resolve class imbalance that helps to detect small objects like license plates. Also, RetinaNet is better faster and requires low computational power. It is more appropriate for real time scalable cloud deployments.

For PERCS implementation, RetinaNet is a better choice because its higher inference speed and lower computational cost. However, Accuracy is considered as the top priority because people trust the accuracy of PERCS. Additionally, License plates are used for law enforcement rules and represent these license plates as evidence in the court.

Ultimately, considering accuracy as our top priority, Faster R-CNN is selected with sufficient cloud resource

3.4 Machine Learning Operations (MLOps)

MLOps integrates DevOps, Machine Learning, and Data Engineering (Kreuzberger et al., 2023). This is mostly focused on cloud environment because of high GPU usage to train ML Models (Pölöskei, 2021).

Transition to MLOps (Subramanya et al., 2022) includes script-based workflows to automated, scalable and reproducible ML pipelines. When the process runs manually, it goes through data preprocessing, model training, evaluation, deployment and monitoring as separate phases. These manual processes could not be completed successfully without human intervention. At this point, MLOps comes to the front for these challenges. By applying DevOps principles, machine learning workflows can assist from more automated data ingestion, model versioning, continuous integration and deployment (CI/CD), and real-time monitoring. Also, Tools like Kubernetes (Edwards et al., 2019), Docker ("Modern App Architecture for the Enterprise," n.d.), and CI/CD pipelines (GitHub, 2024) can be used to ensure models are deployed effectively and automatically retrained when performance declines. This transition enhances scalability, reproducibility, and collaboration, while reducing operating costs and ensuring that machine learning models remain stable, reliable, and production ready.

3.4.1 Overview of MLOps

The MLOps consists of ML Models and software development processes as Figure 4 (wazir et al., 2023). Also, DevOps enables automated installations and continuous monitoring for ML Models. To reach corporate objectives of ML Ops operat ions, it should be capable for providing a collective, constant, repeatable, validated and monitored environment. And Model, Codes and Data are the main three components of the ML Ops development process, and it requires technical automation to run these development phases according to the MLOps structures (*Ml-ops.org*, 2025). MLOps will decrease risks by increasing accessibility and speed in software development because of Quick model construction, high-quality

ML models, speedy placement, and manufacturing. CICD's extensibility enables us to manage and filter many models using the MLOps methods. Similarly, better collaboration among the teams helps to prevent conflicts and transfer the process smoothly with reducing risks.

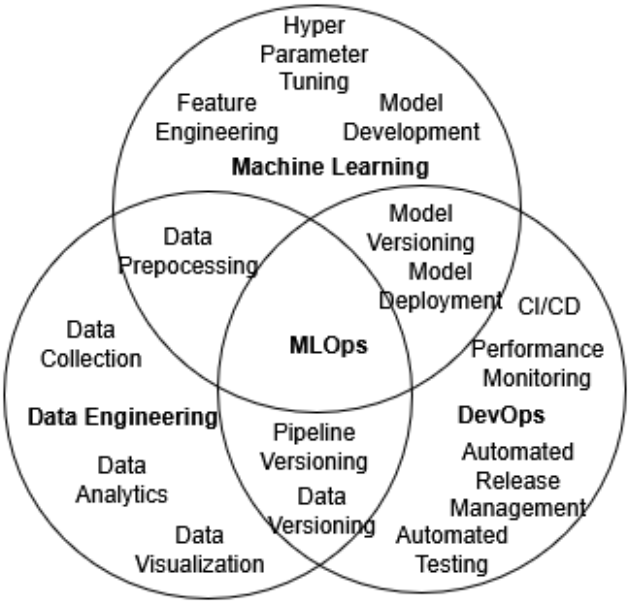


Figure 3. Overview of ML Ops
Figure created by author, based on Wazir et al. (2023).

3.4.2 Machine Learning Lifecycle in MLOps

MLOps based Machine Learning Lifecycle defines as a combination of traditional ML workflows with CI/CD practices. This engagement delivers not only models are trained and deployed effectively but also, they are monitored, maintained, and re-trained efficiently.

The lifecycle steps can be defined as follows:

- **Data Collection**
  MLOps enables automated data pipelines to gather data from different data sources. Also, data pipelines are responsible for data ingestion, data cleaning, transformation, and storage. So, they ensure high quality data

availability for training. Human intervention can be avoided for feature engineering and selection by using automated tools like feature stores.

- **Model Development**
  MLOps enhance this phase using automated model training and evaluation pipelines, ensure reproducibility and consistency. Data engineers or data scientists experiment with different algorithms, tuning hyper-parameters and evaluating performance metrics. Also, different versions of the model are tracked by version control tools like MLFlow (MLFlow Documentation, n.d.) or DVC (Data Version Control · DVC, n.d.). While using automated hyperparameter tuning technique helps to improve performance.

- **Evaluation and Deployment**
  MLOps ensure the smoothness of model transition from development to production which is not like manual ML. CI/CD pipelines incorporate with deployment process and automate it. Model artifact can be deployed into a container or cloud-based location such as Azure Function (Ggailey, n.d.) , Kubernetes Service (Edwards et al., 2019), Amazon Sage Maker (The Center for All Your Data, Analytics, and AI – Amazon SageMaker – AWS, n.d.), etc. After that, the ML model as an API or a microservice can access from other applications.

- **Monitoring**
  MLOps use real-time monitoring tools to model performance (Breck et al., 2017). It notifies the development team when it detects a model drift. Always, the model's accuracy is ensured by an automated retraining pipeline. Other supporting processes like logging, observability, and automated rollback mechanisms support keeping the model stable and dependable.

- **Optimization and Retraining**
  MLOps enable automated retraining cycles, model retrains periodically when the new data is available.

  This iterative process makes sure that machine learning model continues to deliver high performance in real-world applications.

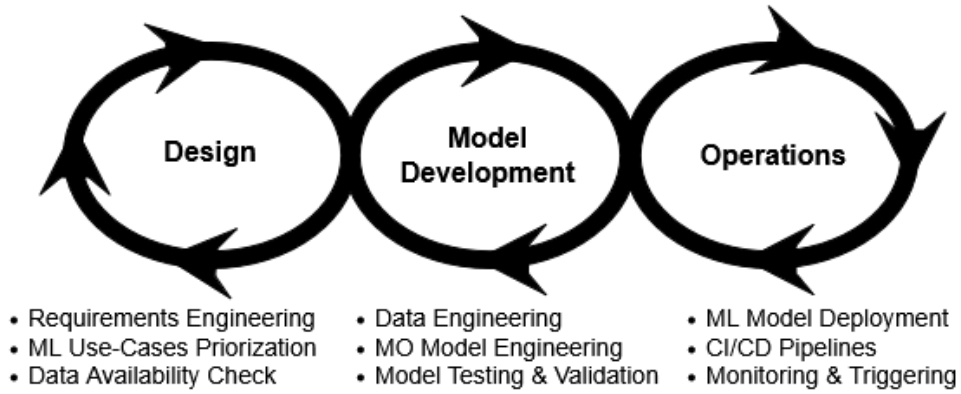


Figure 4. MLOps Lifecycle

Figure created by author, based on Ml-ops.org. (2025).

4 METHODOLOGY

In this chapter, describe how the research implements step by step to choose a suitable surveillance camera, develop the ML model for license plate detection. Mainly, preparing dataset, preprocessing, model selection, training and evaluation phases will be discussed.

4.1 Identify the most suitable Surveillance Camera.

Surveillance hardware plays a critical role in real time vehicle detection. Also, It is the hardware replacement for the old VRS system within the PERCS system, and the chosen hardware should ensure accuracy, responsiveness, and overall performance of the detection system. After evaluating the existing camera options from Milesight (MileSight, 2024), Hikvision (Hikvision Global - Leading the Future of AIoT, n.d.) and Dahua (Dahua Technology - World Leading Video-Centric AIOT Solution & Service Provider, n.d.), the Milesight 4G enabled IP surveillance camera was selected for this purpose. This section explores how the Milesight 4G camera was selected over the three market-leading competitors.

4.1.1 PERCS Requirement

PERCS needs several functional and technical requirements as follows:

- Durability – As an outdoor mounted camera, it should be weather-resistance and support to night vision.
- High Resolution Image capture – Support to identify vehicle accurately.
- Compatible for standard messaging protocol – Prefer to MQTT protocol.
- Remote connectivity – Useful for deployment in non- Wi-Fi locations.
- Sustainability – Efficient in energy and data.

In accordance with the above criteria, three camera models were systematically evaluated. They are Milesight 4G Solar-Powered Traffic Sensing (MileSight Network Technology Co., Ltd., www.milesight.com, 2025), Hikvision DS-

2XS6A46G1-IZS (*DS-2XS6A46G1-IZS/C36S80*, n.d.) and Dahua SD2A400HB-GN-AGQ-PV-SP-EAU (*SD2A400HB-GN-AGQ-PV-SP-EAU*, n.d.)

4.1.2 Comparison of Cameras for Object Detection.

Basically, Milesight and Dahua are more affordable than Hikvision solar cameras. Also, they are using different AI-based techniques for basic vehicle detection. Hikvision uses deep learning LPR models to be whitelisting, blacklisting and vehicle attribute detection and Hikvision provides build-in AI and ColorVu (ColorVu, n.d.) technologies with their solar cameras.

Table 4-1 Feature Comparison
Adapted from Milesight (2024), marcomweb (n.d.), DS-2XS6A46G1-IZS/C36S80 (n.d.), netcamcenter. (2024), shopdelta (n.d.) and Dahua Technology (n.d.).

Feature	**Milesight**	**Hikvision**	**Dahua**
Cost	Budget-Mid	Expensive	Budget-Mid
Solar + 4G Kit	All in One	Modular	Modular
Easy of Deployment	High	Medium	Medium
Weather Resistance	YES	YES	YES

All the above products are suitable for outdoor use, and they are properly prepared to resist weather. According to these three products, Hikvision solar camera is the expensive and premium quality product. Both other products are cost effective, and they offer good balance of performance and affordability (Budget-Mid). While these two products are the most suitable for the PERCS purpose, they fall between the most affordable and the most premium options

Table 4-2.
Technical Comparison of Cameras
Adapted from Milesight (2024), DS-2XS6A46G1-IZS/C36S80 (n.d.), and Dahua Technology (n.d.).

Feature	**Milesight 4G Solar-Powered Traffic Sensing Camera**	**Hikvision DS-2XS6A46G1-IZS**	**Dahua SD2A400HB-GN-AGQ-PV-SP-EAU**
Resolution	2 MP (1920 x 1080)	4MP	4MP
Power Supply	Integrated solar panel + Lithium Battery	80W solar panel + 360Wh lithium battery	5.5W solar panel + 10,000mAh lithium battery
Connectivity	4G LTE + Wi-Fi	4G LTE	4G LTE
Protocol Support	MQTT, HTTP, RTSP, ONVIF	HTTP, RTSP, ONVIF	HTTP, RTSP, ONVIF
Waterproof Rating	IP67	IP67	IP66
Installation	All-in-one plug & play solar unit	Modular setup with external solar components	Integrated compact PTZ solar design
Ideal Use Case	Smart traffic monitoring, off-grid vehicle detection	Remote border and highway surveillance	Budget surveillance with motion alerts

Resolution is a key attribute to license plate detection. Higher resolution of an image helps to capture fast moving vehicle details in different light conditions. But in PERCS implementation, vehicles slow down near to the parking gate therefore high resolution is not an essential requirement. 2MP is more sufficient which is offered by Milesight. However, Hikvision and Dahua provide 4MP resolutions and they would be highly recommended where precision is a critical requirement.

The cameras require not only battery storage but also solar power for remote deployments. The Milesight camera designs a compact unit including its solar panel and batteries. However, Hikvision facilitates the heaviest solar solution, as 80W solar panel with 3600Wh Battery, which is more suitable for heavy workloads with limited sunlight. Dahua designs light-duty power solution with 5.5W panel and 10000mAh battery. Based on these power solutions, Hikvision leads a robust power supply for their cameras.

All three cameras offer 4G LTE connectivity and it helps to deploy these devices in locations that are both physically isolated and lack of traditional infrastructure such as electricity and wired internet. Also, Milesight is additionally supporting to WIFI which is adding flexibility during the installation and maintenance.

These camera models support HTTP, RTSP and ONVIF protocols, but only Milesight includes MQTT support natively. Especially, Realtime messaging with edge computing devices is very useful to run a cloud-based vehicle system smoothly. This enhances Milesight's value by increasing its integration potential.

According to the IP rating (IP Ratings, n.d.), which ensures the durability of each device. Milesight and Hikvision meet the IP67 standard, and it offers total protection from dust, rain and different weather conditions. Dahua’s rating behind the others as IP66, slightly less protection than other two products.

Furthermore, Device installation is a major operational consideration especially in isolated areas. The Milesight camera is an all-in-one design, which makes life easy

to deployment and setup time. On the other hand, Hikvision increases complexity and expense with modular installation. Also, Dahua's design offers a balanced product between simplicity and functionality, but it may lack of durability and expandability.

Based on the ideal use cases of each camera and the above factors, Milesight 4G Solar-Powered Traffic Sensing Camera can be taken as the most closely aligned device for the purpose of vehicles detection in the PERCS.

4.2 Camera operation in PERCS

Milesight 4G solar-powered traffic sensing camera provides two major image capturing options that are continuous capturing and trigger capturing. Continuous capture occurs at set intervals, regardless of whether an object is detected or not. But trigger capture occurs only when an object detects. Trigger capture is the most suitable option to keep detected objects and send them to the PERCS system (Milesight, 2022). (Camara will be set up aside from the entrance or exit lane and it will be configured to capture an image when an object is detected.)

Once captured images receive by the PERCS, they send to another third-party service which is called REKOR (RekOr - Delivering Revolutionary Roadway Intelligence, n.d.). That service extracts those vehicles details (ex: Vehicle Number, Colour, State, etc.) and return them to PERCS. Since the REKOR service has a monthly request limit, submitting an image that does not contain a vehicle license plate will result in an empty response. It is an unnecessary cost because more vehicles in busy days are captured by the camera without license plates in the images.

4.3 Dataset

In Computer vision, dataset helps ML models to learn complex patterns and variations in the development, training and evaluation phases. The quality of the dataset, size and the diversity heavily affect to the performance of ML models. Con-

sidering the PERCS parking scenario, vehicle images should be captured under varying weather conditions and different background settings, including both daytime and nighttime. On the other hand, some of large-scale datasets like ImageNet (Deng et al., n.d.) are playing a huge role in deep learning research with providing thousands of categories, millions of labelled images to enhance the accuracy of both object detection and image classification models. Furthermore, some task-specific datasets like COCO (T. Lin et al., 2014) and PASCAL VOC (Everingham et al., 2009) facilitate comprehensive annotations for complex vision tasks. Without well-designed dataset, it can be poorly performed as a biased, overfitted Model. Therefore, it is very important to construct or choose the right dataset in any computer vision task for real world performance.

4.3.1 Dataset Selection.

As explained in 4.2, vehicle images are captured by 4G solar camera, and they send to the PERCS system through a message broker (Chapter 5 – Deployment and Real-Time Testing). These images are used to train License Plate detection model.

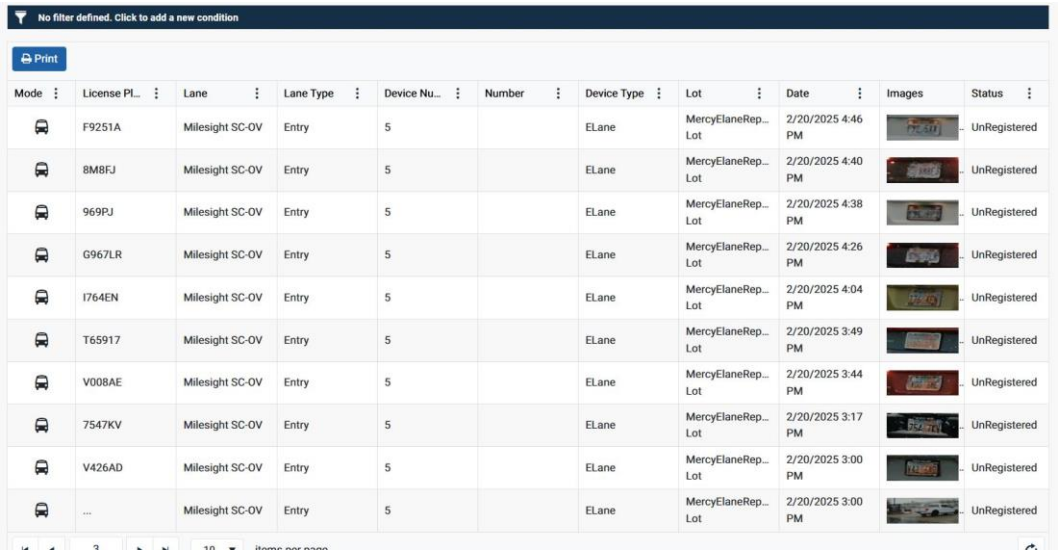


Figure 5. PERCS Vehicle Inventory

4.3.2 Data Annotation

Label Studio (HumanSignal, n.d.) has used to mark boundary boxes for license plates. License plates are focused to validate the visibility of license plate of vehicles for Rekor API (RekOr - Delivering Revolutionary Roadway Intelligence, n.d.).

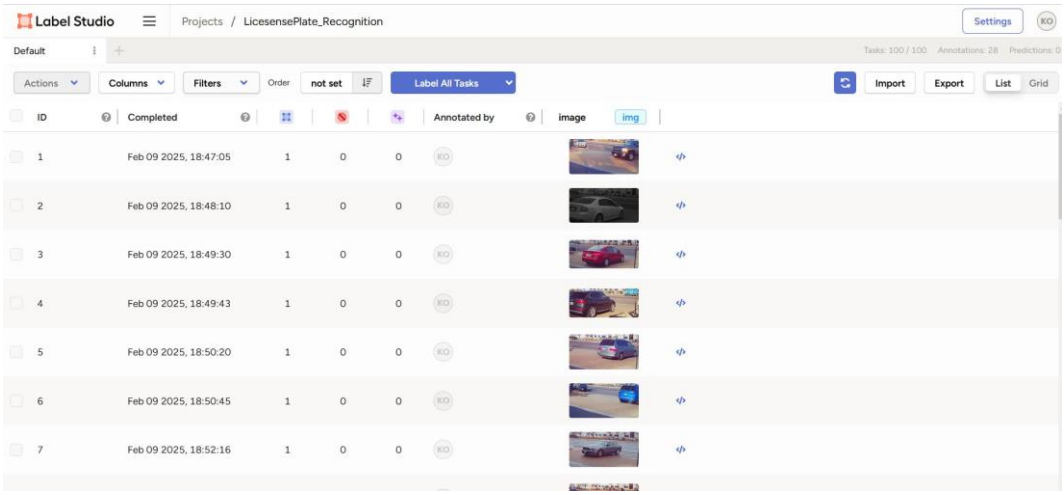


Figure 6. Vehicle Images in LabelStudio

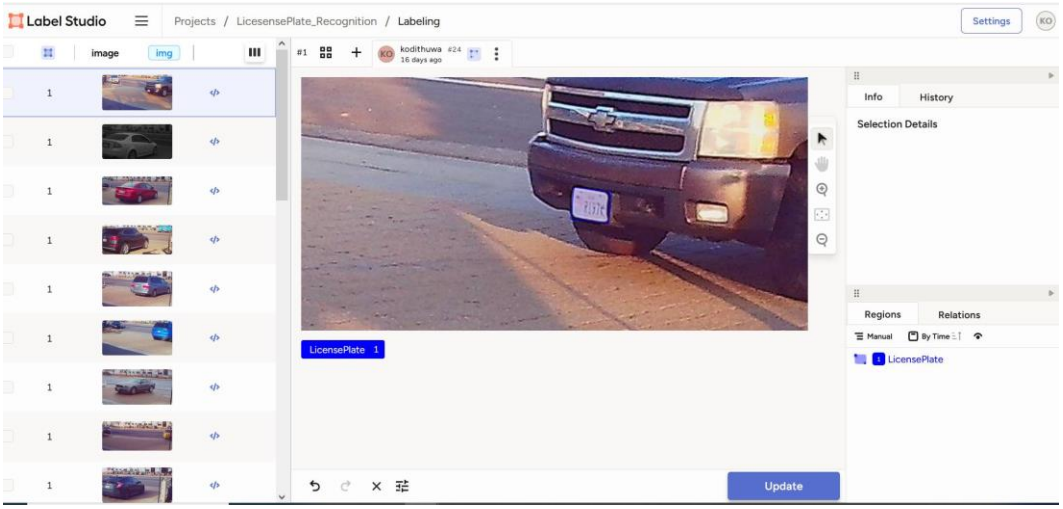


Figure 7. Annotate Boundary Boxes

4.3.3 Data Preparation

In this phase, data change to required format to model training purpose. For this transformation pipeline, Albumentations library (Buslaev et al., 2020) is applying series of augmentations for images and maintain their bounding boxes in COCO format (Lin et al., 2014). As a practice, resizing shortest side or the longest side of images depends on the specific task and hardware constraints. Both approaches are used in the industry, but they address different purposes. (Ren et al., n.d.)

Table 4-3. Effects of Image Resizing

	Advantages	**Disadvantages**
Resize Shortest Side	• Image is preserved. • No Distortion • Control Scale of the image • Useful when dataset consists of images with different aspect rations.	• Longest side becomes large. • Lead to memory bottleneck or Long Training Time.
Resize Longest Side	• Image fits withing the memory constraints of the GPU. • More suitable for models with hardware limitations. • Prevents excessive large images. • Reduce memory consumption and training time. • Effective when larger size of images in the Dataset. • The objects remain sufficiently visible without compromising memory.	• Shortest side becomes smaller. • Details might be lost. • Not visible enough smaller objects for detection

Also, Milesight camera produces images with same dimensions as 1392*576px. In Faster R-CNN process, Image dimensions are resized during image pre-processing for optimal performance.

Considering Maximum dimensions constraint. The longest side is changed as 1000px to avoid excessive memory consumption (Ren et al., 2015). The longest side (1392px) changes as 1000px, the new dimensions would be:

New Width = 1000px

Current Aspect Ratio = 11392/576 = 2.42

New Height = 1000/2.42 ≈ 413px

So, the resized image size would be 1000*413px.

While it uses 30% horizontal and vertical flip to orientation change and, altering brightness contrast and saturation are changed by 10%. Finally, images are converted to PyTorch tensors which is used for deep learning models.

```python
def get_transforms(train=False):
    if train:
        transorm = A.Compose([
            A.LongestMaxSize(max_size=1000), # Resize the input image to 413x1000
            A.PadIfNeeded(min_height=413, min_width=1000), # Add extra padding if needed
            A.HorizontalFlip(p=0.3), # Horizontally flip the image with a probability of 30%
            A.VerticalFlip(p=0.3), # Vertically flip the image with a probability of 30%
            A.RandomBrightnessContrast(p=0.1), # Randomly change brightness and contrast of the image with a probability of 10%
            A.ColorJitter(p=0.1), # Randomly change the brightness, contrast, and saturation of an image with a probability of 10%
             A.Normalize(mean=(0.485, 0.456, 0.406), std=(0.229, 0.224, 0.225)),  # Normalize
            ToTensorV2() # Convert the image to PyTorch tensor
        ], bbox_params=A.BboxParams(format='coco')) # Define the format of the bounding boxes
    else:
        transorm = A.Compose([
            A.Resize(413, 1000),
            ToTensorV2()
        ], bbox_params=A.BboxParams(format='coco'))

    return transorm
```

Figure 8. Transformation Pipeline

4.3.4 Faster R-CNN Architecture

Faster R-CNN, short for Faster Region-based Convolutional Neural Network, operates as a two-stage object detection framework. Breakdown of the architecture as Backbone and Region Proposal Network as follows:

Backbone

Backbone is a convolutional neural network (CNN) which is used to extract feature maps from the input images. CNN produces feature map as an output. This feature map uses for region proposal and object classification.

Table 4-4. Recommended backbone for Vehicle Detection

Backbone	Architecture Type	Pros	Cons
ResNet-50	Deep Residual Network (50 Layers)	Residual learning assists for deeper networks.	Require More Computation.
ResNet-101	Deeper Residual Network (101 Layers)	Higher accuracy than ResNet-50	Slower inference speed.
ResNeXt-101	Wider Residual Network (grouped convolutions)	Effective feature learning, better accuracy.	More expensive computation.
MobileNetV2	Light-weight CNN for mobile applications	Fast Inference, Low memory usage.	Lower Accuracy.

ResNet-50 is a well-balanced model with accuracy and speed. Also, it provides good accuracy. While ResNet-101 uses more layers, and it enables to detect small and occluded vehicles, But ResNet-101 is slower than RestNet-50. On the other hand, MobileNetV2 which is introduced for constraint devices like embedded systems, mobile devices, or low-power applications. Main differentiation of MobilNetV2 is lightweight and fast but lower accuracy than other two models. (Elharrouss et al., 2024)

```
# Load the faster rcnn model
model = models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
in_features = model.roi_heads.box_predictor.cls_score.in_features # number of input features for the box predictor
model.roi_heads.box_predictor = models.detection.faster_rcnn.FastRCNNPredictor(in_features, n_classes)
```

Figure 9. Configure ResNet-50 for Faster R-CNN

Region Proposal Network (RPN)

RPN (Ren et al., n.d.) responsible to generate region proposals. It marks bounding boxes where objects might be potentially existed. The RPN creates a set of proposals as outputs with high objectness score.

Objectness Scores – An object availability in a region .

Bounding box coordinates – adjustments to anchor boxes to fit with objects.

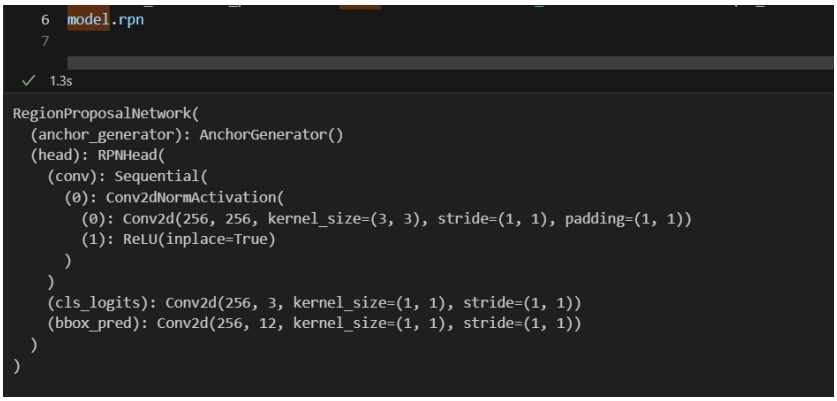


Figure 10. RPN Module

ROI (Region of Interest) Pooling

ROI pooling (Singh, 2024) converts different size of proposals generated by RPN to a fixed-size feature map (eg: 8 x 8). This uses max pooling to manage the most important spatial information. It takes an object proposal from RPN and maps the proposal onto the feature map from the CNN backbone. Then after, divide into a fixed number of grid cells. And perform max pooling.

ROI Head

After completion in ROI pooling, the ROI head processes the fixed-size feature maps to classify objects as License plate and Background.

4.3.5 Model Training

As the first step, prepare the dataset using images and their bounding box annotations. COCO format is used for annotation mappings, and each bounding box is having an associated class and defined two classes (Background and LicensePlate) for these annotations.

Initially, annotate 500 images and they split for Training 400 images (80%), Validation 50 images (10%) and Testing 50 images (10%). This way, ensure the model learns well where there is having enough unseen data in evaluation.

Also, fasterrcnn_resnet50 backbone is used as a feature extractor for this purpose. Because Resnet 50 is a well-balanced with better accuracy and the fastest one in Resnet family.

```
# lets load the faster rcnn model
model = models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
in_features = model.roi_heads.box_predictor.cls_score.in_features # number of input features for the box predictor
model.roi_heads.box_predictor = models.detection.faster_rcnn.FastRCNNPredictor(in_features, n_classes)
```

Figure 11. Resnet 50 Backbone

Runs 10 epochs (iterations) and observe continuously decreasing object error loss as below.

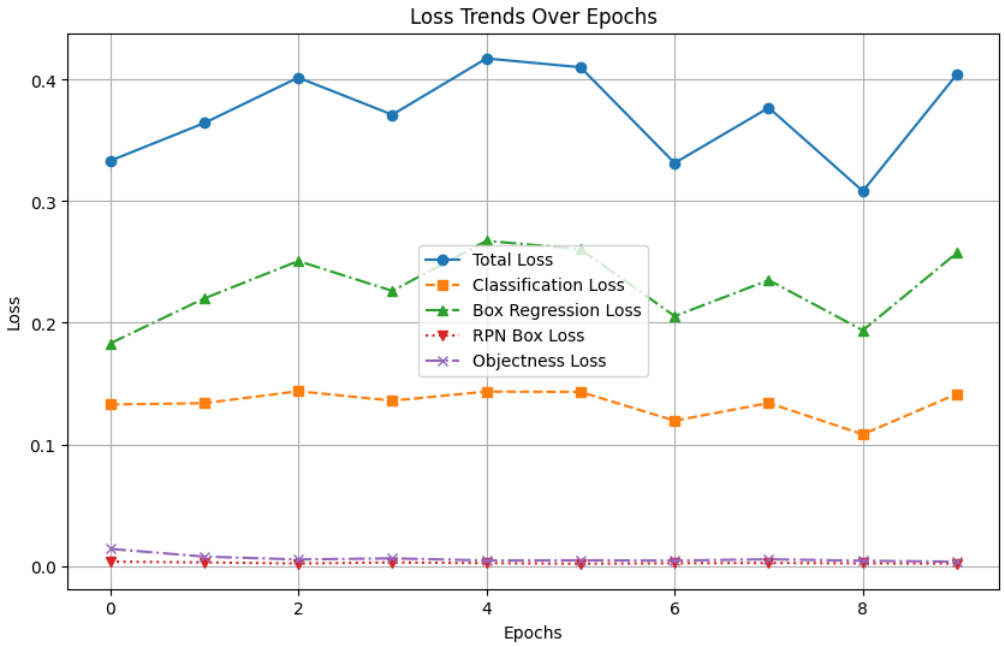


Figure 12. Loss Trends Vs Epochs

As the above charts, Minimum **Total Loss** occurred in 8th epoch. It can be taken efficient result after model trains 8 epochs.

4.3.6 Model Evaluation

Mean Average Precision (mAP) (Hui, 2020) is the most reliable metric to evaluate object detection models such as Faster R-CNN. it is using precision and recall calculating how well the model detects and localizes object.

```python
# Define metric
metric = MeanAveragePrecision(iou_thresholds=[0.5, 0.75])  # COCO-style IoU

# Evaluate Model on Test Dataset
with torch.no_grad():
    for images, targets in test_dataset:
        images = [img.unsqueeze(0) if len(img.shape) == 2 else img for img in images]
        outputs = model(images)  # Run inference

        # Convert predictions to required format
        predictions = [{
            "boxes": outputs[0]["boxes"].detach().cpu(),
            "scores": outputs[0]["scores"].detach().cpu(),
            "labels": outputs[0]["labels"].detach().cpu()
        }]

        # Convert ground truths to required format
        ground_truths = [{
            "boxes": targets["boxes"],
            "labels": targets["labels"]
        }]

        # Update metric
        metric.update(predictions, ground_truths)

# Compute final mAP
result = metric.compute()
print("mAP@0.5:", result["map_50"])
print("mAP@0.5:0.95:", result["map"])
```

Figure 13. Calculate mAP using Python

```
mAP@0.5: tensor(0.8119)
mAP@0.5:0.95: tensor(0.6291)
```

Figure 14. mAP Results.

mAP@0.5 (81.19%) → Good object detection but it might have some unclear bounding boxes.

mAP@0.5:0.95 (62.91%) → Stricter evaluation, showing how well the model localizes objects with high precision.

This could be fine-tuned or enhanced more with changing better backbone like ResNet 101 with more training epochs.

At the end of the model training, Evaluate the model using different images as below.

Figure 15. Testing Vehicle Image 1

Figure 16. Testing Vehicle Image 2

5 DEPLOYMENT AND REAL-TIME TESTING

As a major enhancement of the PERCS system, PERCS authority engage with Milesight security solutions. Milesight (MileSight, 2024) is a technology company that specialized in the development and manufacturing of advanced video surveillance solutions and Internet of Things (IoT) products. The company is known for producing high quality, innovative products that cater to a wide range of industries, including security, smart cities, transportation and more.

5.1 System Overview

PERCS is a feature rich parking system. It is evolving day by day for its customers satisfaction. The PERCS system has developed as a cloud-based system. All administrators and users use this cloud based centralized system. The system is intelligently handled users' accessibility according to user's role.

PERCS has already given below features for its clients.

- **Automated Ticketing**: The system allows for the automatic issuance of parking tickets through various methods, such as handheld devices, vehicle-mounted cameras, or stationary cameras. This reduces the need for manual intervention and increases enforcement efficiency.
- **Real-time Data Integration**: PERCS systems often integrate with other databases, such as vehicle registration records or payment systems, to provide real-time data on parking violations, payments, and enforcement actions.
- **Revenue Collection**: The system can streamline the process of collecting fines and fees, offering multiple payment options for violators, such as online payments, mobile apps, or kiosks.

- **Reporting and Analytics**: PERCS typically includes tools for generating detailed reports and analytics on parking enforcement activities, revenue collection, and compliance rates. This data can be used to optimize enforcement strategies and improve overall efficiency.
- **Integration with Smart Parking Systems**: Some PERCS solutions are integrated with smart parking technologies, such as sensors and automated license plate recognition (ALPR) systems, to further enhance the accuracy and efficiency of parking enforcement.

Also, the Milesight integration will directly affect for **Integration with smart parking systems**.

5.2 Architecture Design of Milesight Integration

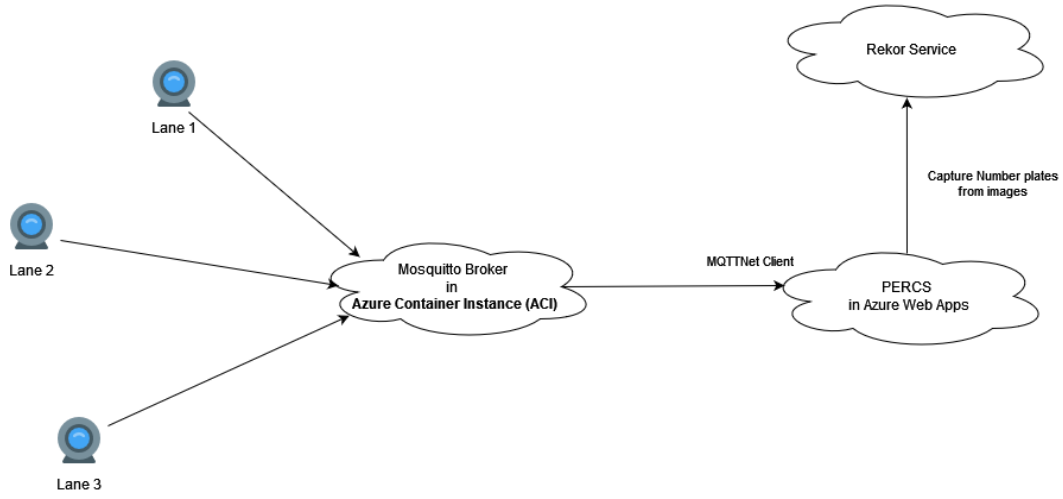


Figure 17. Overview of Milesight Surveillance Cameras Integration

Surveillance cameras for the selected lanes define in the PERCS as IOTLane devices. As the first step, setup the Mosquitto MQTT broker (Benevides, 2022) in the Azure Environment. Afterward, each camera configures individually with using MQTT broker details and setup the snapshot publishing settings as well.

In the PERCS-End, MQTTNet client (Dotnet, n.d.) will catch the snapshot detail of each lane and send them to Rekor - OCR (RekOr - Delivering Revolutionary Roadway Intelligence, n.d.) service to validate license plate. Once it verifies the license

plate, the response of the Rekor (State, Make, Model, Colour, Year) will be stored in PERCS database.

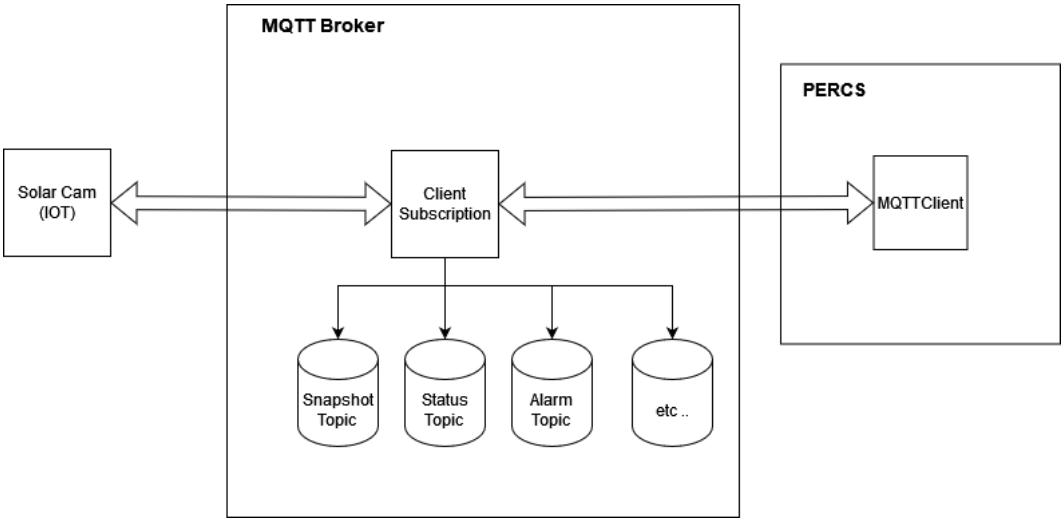


Figure 18. Pub/Sub Integration with MQTT Broker

Milesight surveillance camera is an intelligent device. it detects objects, capture snapshots and send it to MQTT Broker using MQTT Topics. Also, it contains a re-chargeable battery to store energy for use during low sunlight or night-time. It enables to transmit video feeds and other data without a wired network connection and easy to install any area even with limited infrastructure. It produces high-definition videos up to 4K or higher. This makes clear and details footage even in outdoor environments. Some models contain edge processing capabilities as processing video and save in SD card for local recording.

5.3 Proposed License Plate Detection Model Integration

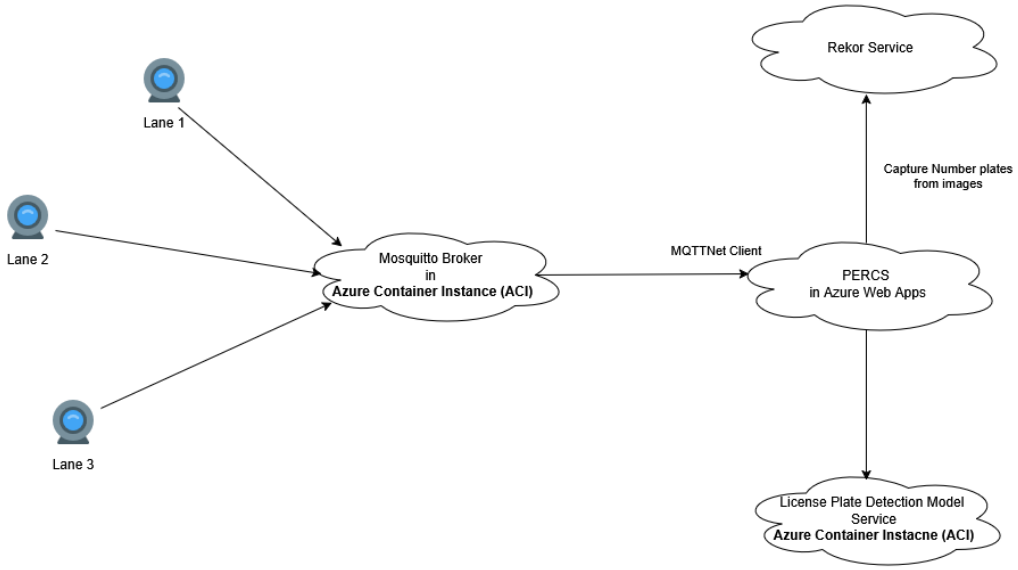


Figure 19. ML Model in Azure Container Instance (ACI).

New Machine Learning model deploys as an Azure Container Instance (Tomvcassidy, n.d.) in MS Azure Environment. According to the PERCS requirement, Azure container instance is the most suitable solution because ML inference service is a simple containerized API without infrastructure management and easy to deploy for a small-scale solution.

5.4 Data Flow from Edge Device (4G Solar Camera) to MQTT Broker

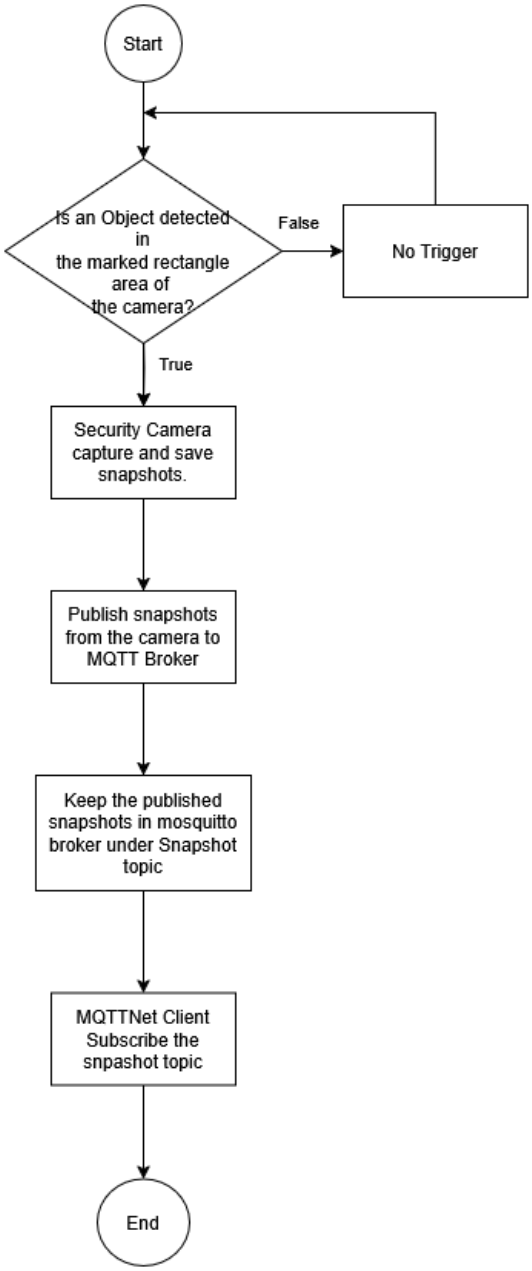


Figure 20. Data Flaw Diagram from Camera to MQTT Broker

Milesight 4G surveillance camera is triggered when an object detects in the marked squared area, and it saves in the camera While simultaneously publishing event to the MQTT broker. After that, image saves in the broker under the snapshot topic.

5.5 Recording data in the PERCS

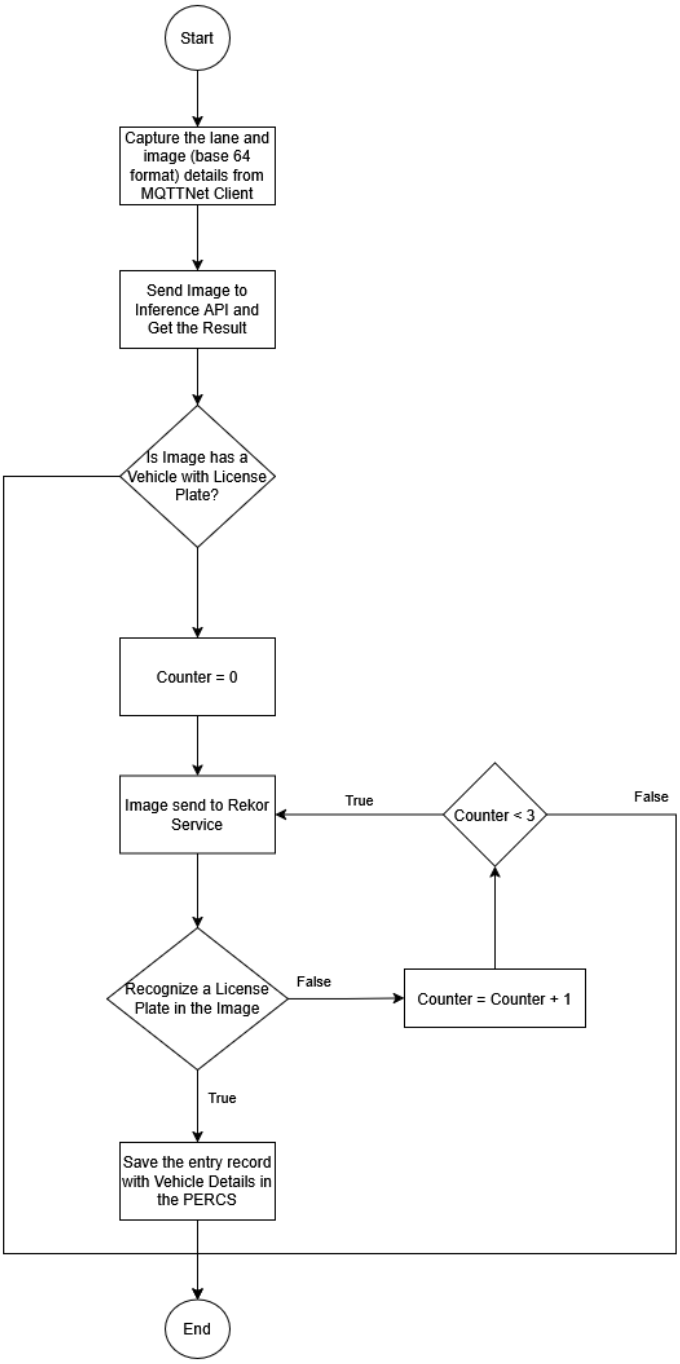


Figure 21. Image Saving Algorithm in PERCS Service.

Firstly, MQTTClient (Dotnet, n.d.) in the PERCS system captures subscribed records from MQTT broker (Light, 2017) and then after, the image is sent to Machine Learning Model through the Inference API to check whether license plate of vehicle is visible or not. If the license plate of vehicle is detected in the ML service, it will be sent to the Rekor service (RekOr - Delivering Revolutionary Roadway Intelligence, n.d.) to extract vehicle details.

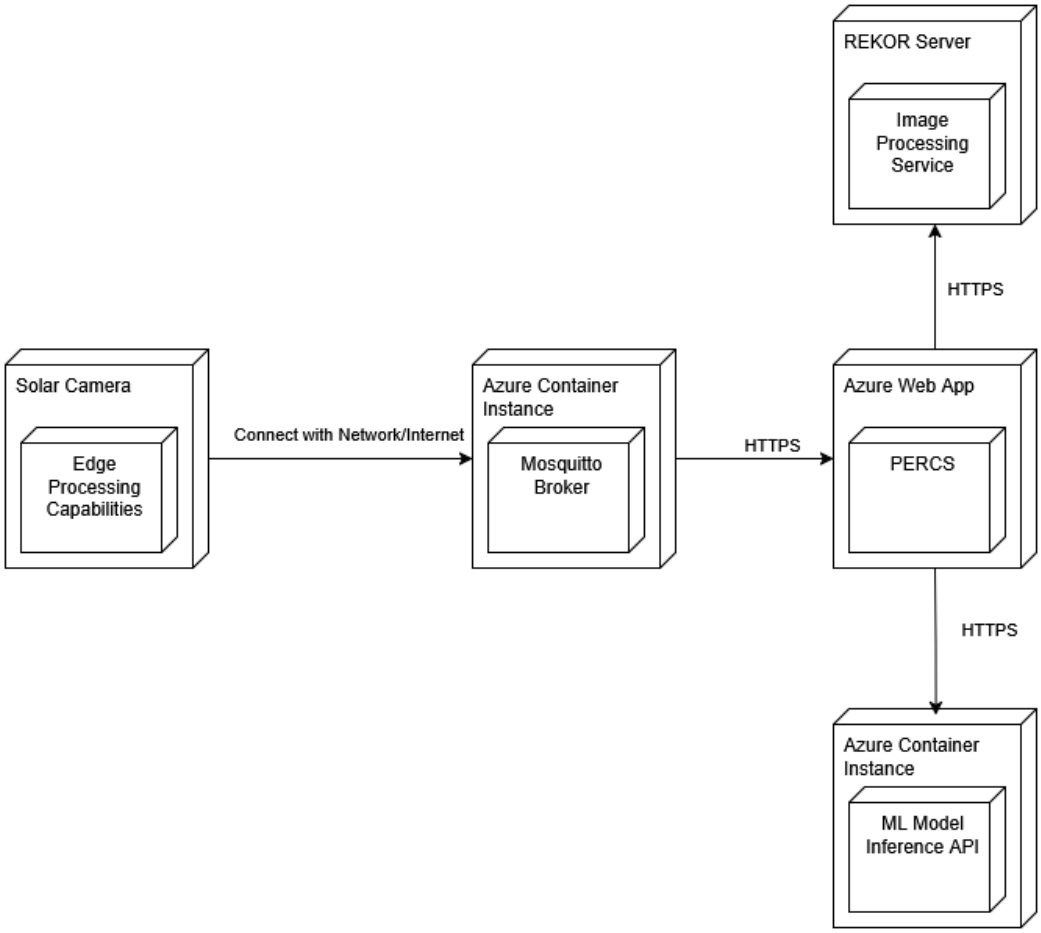


Figure 22. System Deployment Diagram

5.6 Tools and Frameworks

There are some specific tools, libraries have been used for this extension development. Brief introduction of each component of them included in this section. It will be helpful to get a general idea of each of them individually.

5.6.1 Mosquitto MQTT Broker

Mosquitto (Light, 2017) is a lightweight and open-source software. It has been designed for limited resources such as micro controllers or embedded systems. It supports MQTT (Usmani & Frankfurt University of Applied Sciences, 2021) Version 3.1, 3.1.1 and 5.0. it can handle thousands of concurrent clients and large volume of messages. Mosquitto has been recommended for small or large scale IoT deployments. It is also easy to integrate with web applications because it supports for web sockets and allow to send MQTT messages over HTTP.

In this purpose, Mosquitto deploys an azure container instance from a docker image. Also, port should be exposed to communicate with clients. Its default MQTT port is 1883. Once configured the Mosquitto broker in the cloud, Username and password of the broker can be configured (Benevides, 2022b).

5.6.2 MQTTnet Client

MQTTnet (Dotnet, n.d.) is a MQTT protocol based high-performance .NET library for communication, and it supports MQTT protocol up to version 5. The library available MQTT Server (Broker) and MQTT Client. Most of .NET Framework versions and CPU architectures are supported to MQTTnet library.

- Consist of ManagedMqttClient maintains connection and subscriptions automatically.
- Consist of core LowLevelMqttClient with low level functionality.
- TCP (+TLS) or WS (WebSocket) protocols are supported.
- Compatible with Microsoft Azure IoT Hub

- Rx support (via another project)
- Messages from applications are queued and re-scheduled for higher QoS levels automatically.

5.7 Scalability of the system.

As system demand increases when the business grows, it becomes essential to scale both horizontally and vertically to handle the rising workload without compromising performance. A scalable system can be managed by adding resources such as servers or storage, allowing it to adapt efficiently.

Also, a scalable system performs as a sustainable and forward compatible solution because required resources are only allocated to implement the system. In this development, following attributes promote the scalability of the system.

Modular Architecture: Modular approach to design (camera and cloud services), they can be scaled independently as needed.

Edge Computing: Process data locally on the 4G surveillance Cameras. While this reduces the amount of data sent to the cloud and minimizes latency, helping the system handle more devices efficiently.

Efficient Data Protocols: MQTT is a lightweight and efficient communication protocol that is designed for IoT environments. This protocol helps in managing bandwidth and ensuring timely data transmission.

Cloud Services: Mosquitto, ML Model and PERCS are hosted in Azure cloud services that can allocate resources on demand wise dynamically.

Distributed Systems: MS SQL server, Web apps and Azure instance containers are distributed systems and that can handle large volumes of data and scale horizontally by adding more.

5.8 System Performance Considerations

Performance is a major concern for a software system which is including response time, throughput, resource utilization and scalability (discussed in section 5.9). A well-performed system should respond to an end users request quickly, and it is able to handle multiple requests at once with minimum delay. Also, it should be maintained consistent performance in the different workloads.

After Milesight Camera integration, the existed system architecture has been changed. Also, data flow of the system should be tested like how to mosquito broker behave in the system and how long it takes to response by ML Model Inference API.

Table 5-1. Mosquitto Broker Performance Considerations

Message Throughput	**Benchmark**: Measure the number of messages per second the Mosquitto broker can handle. **Consideration**: The performance of the broker depends on the CPU, memory, and network throughput allocated to the Azure Container Instance. Start with a smaller configuration and benchmark performance under load. **Tuning**: Increase the maximum connections (max_connections), optimize message queue sizes, and configure MQTT keep-alive intervals to balance load and latency.
Latency	**Benchmark**: Measure the time spends for a message is to be delivered to the broker from the IoT device and then to the WebApp.

	Consideration: Network latency between IoT devices and the Mosquitto broker is critical for real-time IoT applications. Use low-latency configurations in Azure regions close to the edge devices.
Connection Handling	**Benchmark**: Test the number of simultaneous connections the Mosquitto broker can manage. **Consideration**: IoT systems with thousands of devices require efficient handling of MQTT connections. Azure Container Instances need to be scaled (CPU, memory) based on the connection load.
Message Retention and Persistence	**Benchmark**: Test the message persistence and delivery guarantees when the broker goes down or when devices reconnect. **Consideration**: The persistence feature in Mosquitto allows the broker to retain messages for clients that are offline. Ensure proper disk I/O allocation if persistence is used.

Table 5-2. Azure Container Instances (Broker, ML Model) considerations.

Resource Allocation	**Benchmark**: Measure CPU and memory utilization under different loads (number of devices and message rates). **Consideration**: Ensure the container instance is provisioned with adequate resources for peak loads. For high

	performance, scale CPU and memory as needed, or consider deploying Mosquitto on Azure Kubernetes Service (AKS) for horizontal scaling.
Autoscaling	**Benchmark**: Evaluate the performance when scaling up and down container instances based on demand. **Consideration**: Azure Container Instances do not natively support auto-scaling, but you can use Azure Kubernetes Service (AKS) with a load balancer for dynamic scaling.

5.9 Security

As a fundamental aspect of a software or IT infrastructure, considering here to protect services from unauthorized access, breaches and malicious attacks. all the entities are authenticated, and authorization individually as follows:

PERCS Authentication

Verify the effect of PERCS authentication for new integration with surveillance cameras and ML Model.

Mosquitto Authentication

Ensure mosquitto is configured to require authentication for MQTT connections. This can be achieved by using username/password mechanism or certification.

Azure Machine Learning Endpoint Authentication

Use API Key to secure ML Endpoint.

PERCS Authorization

Ensure that PERCS default access control avoid accessing the PERCS APIs to outside.

Mosquitto Access Control

Use Mosquitto's access control lists (ACLs) to define what actions (e.g., subscribe, publish) are permitted for different clients.

5.10 Network Traffic Analysis

Network traffic analysis is the process to monitor data packets that travel across the network with connecting remote entities. It helps to understand the information flow, detect malicious functions, identify patterns and ensure optimal network performance. It is also helpful to mitigate security threats like unauthorized access, malware or data breaches.

In this Implementation, Azure Network Watcher (Halkazwini, n.d.) mainly uses to monitor and analyse network traffic related to PERCS, ML Endpoint and Mosquitto. Look for any unusual patterns or unauthorized access attempts.

5.11 Data Encryption & Compliance

The most essential approach for real-time systems that have personal, financial and business critical information. Data Encryption helps to restrict unauthorized access and cyber threats. On the other hand, compliance involves to legal body and regulatory standards such as GDPR (General Data Protection Regulation (GDPR) – Legal Text, 2024), HIPPA (Rights, 2025) or PCI-DSS (PCI Security Standards Council, LLC, 2018), they promote to use encryption and other security measures to build the trust with customers and stakeholders.

Data in Transit

Ensure that data transmitted between devices and the Mosquitto broker is encrypted using TLS/SSL. Verify that Mosquitto is configured to use strong encryption protocols and that certificates are valid.

Data at Rest

For data stored in PERCS or any other storage solutions, ensure that encryption is enabled to protect data at rest. Also, ensure IoT network complies with relevant security standards and regulations (e.g., GDPR, HIPAA).

6 SYSTEM MAINTENANCE AND MONITORING

Maintaining this system integration consists of several key practices focused on performance, security, monitoring and scalability. It involves update software, applying security patches, optimizing system configuration to avoid issues before they happen. On the other hand, real time monitoring helps to trigger alerts for administrators when a fault happens. Furthermore, it is important to monitor unusual network activities, hardware failures and to address any issues promptly once they detected.

As the outcome of maintenance and monitoring, reducing system down time, extend system lifespan, and ensure a smooth and efficient user experience.

6.1 Monitoring and Logging

There are number of tools and services are used in monitoring and logging in the azure environment. Monitoring tools can be used to continuous observation in system metrics such as CPU , memory, network utilization. Also, logging tools help to record details of system events, user activities and error messages.

Both monitoring and logging together help to identify and resolve problems quickly and support compliance with security and operational standards. Azure environment has given tools to monitoring and logging as follows:

- Azure Monitor (Bwren, n.d.)
  Use azure monitor and Azure container insights to observe performance metrics (e.g. CPU, memory usage) and monitor the health of the Azure Container Instance (ACI) hosting the Mosquitto broker and ML Model.

- Web App Monitoring (Bwren, n.d.)
  Use Application Insights for real-time monitoring of the Azure WebApp, including response times, errors, and user behaviour.

- Enable logging for the Mosquitto broker to capture key events such as authentication attempts, message delivery status, and connection issues.

- Enable logging for the ML model failures, API events and connection issues.

- Centralize log collection using Azure Log Analytics (Austinmccollum, n.d.) to capture logs from both the WebApp and Mosquitto broker and set up alerts for critical security events or performance degradations.

6.2 Setup Alerts

Setting up alerts in Azure Monitor helps to observe performance, health and the availability of resources. With configuring Azure Security Centre and Azure Monitor to trigger alerts for specific issues, such as high CPU usage, memory leaks, or downtime.

Also, it can be notified after setting thresholds for critical metrics (e.g., excessive packet loss, long response times) and configure notifications via email, SMS, or integration with DevOps tools.

6.3 Security Management

Security management helps to confirm the system safety from outside attacks such as traffic of Azure container Instances is restricted and only allow to communicate with trusted IP addresses and secure ports (eg: MQTT over TLS port 8833). Also, firewalls should be configured to restrict unauthorized devices and applications to communicate with the Mosquitto broker or ML API. Further, allocate Mosquitto broker and ML API inside an Azure vNet (Asudbring, n.d.) and use vNet peering for secure communication with PERCS Webapp. This setup never exposes the container instances to public.

Also, TLS/SSL protocols are used to secure communication between IoT devices and the Mosquitto broker. The broker is configured to use a valid certificate. Also,

implement a strong authentication mechanism for devices and applications that are accessing the Mosquitto broker, using either username/password or client certificates.

6.4 Security Audits and Compliance

Perform security audits on the Azure infrastructure, including the WebApp, container instance, and network configurations. Look for common vulnerabilities such as misconfigurations, open ports, and weak access controls.

- Conduct **penetration testing** on both the WebApp and Mosquitto broker to identify and fix security vulnerabilities.
- Ensure that IoT system complies with industry standards and regulations (e.g., HIPAA, GDPR, ISO/IEC 27001). Regularly review security and data protection policies to ensure compliance.

6.5 Incident Response and Troubleshooting

This process involves identifying, analysing and resolving unexpected issues or confusions in the system and the mechanism begins from anomaly detection or real-time failure. After the issue is identified, the team assess the severity and impact of the issue and decide priority actions.

In PERCS scenario, Incident response plan should be developed and maintained for potential breaches or failures. This plan should include steps for detecting incidents, notifying stakeholders, and recovering from security incidents or system outages.

Further, Train the team to follow the response plan and regularly test it with simulated incident scenarios. Also, Team is fully responsible to implement tools and workflows for debugging and troubleshooting issues. Azure Diagnostics extension should be used to collect detailed telemetry and diagnostic logs from both the container instance and WebApp.

7 CONCLUSION & FUTURE WORKS

In this thesis, it has been worked in real-time license plate detection using machine learning. As the first step, it was explored the information about machine learning models and deep learning models, and how they classified purposely in the literature review. Also, working with deep learning models and critically analysis which one is the most suitable for PERCS requirement and compared models, DETR, RetinaNet, SSD, Faster R-CNN, Mask R-CNN and YOLOv8. Select Faster R-CNN model based on research findings. They highlighted the potential of Faster R-CNN for vehicle detection and smart surveillance systems. The model shows real world deployment considering some ground variations such as obstructions, different lighting conditions and computational constraints.

Also, Vehicle images are taken from PERCS which are captured from Milesight 4G surveillance camera. They have converted to COCO format and used to train the Faster R-CNN model.

Finally, Dataset preparation, Faster R-CNN training and evaluation completed using MLOps (Azure ML Studio). It ensures automation, scalability, and reproducibility. Also, MLOps enables smoothly update, monitoring and improving model performance recurrently.

In the future, this is planned to be extended to license plate recognition, which involves detecting, segmenting, and recognizing characters on vehicle license plates using computer vision and deep learning techniques. Since License plate detection has been completed, only need to apply character recognition using Optical Character Recognition (OCR). This way, the REKOR service can be replaced from the PERCS system.

REFERENCES

A Division of Quest Solution Inc. (2019). *VRS for Parking Systems - Overview*. http://www.htsol.com/wp-content/uploads/2019/04/HTS-Parking-Overview-White-Paper.pdf

Alqahtani, H., & Kumar, G. (2023). Machine learning for enhancing transportation security: A comprehensive analysis of electric and flying vehicle systems. *Engineering Applications of Artificial Intelligence*, *129*, 107667. https://doi.org/10.1016/j.engappai.2023.107667

Asudbring. (n.d.). *What is Azure Virtual Network?* Microsoft Learn. Retrieved April 15, 2025, from https://learn.microsoft.com/en-us/azure/virtual-network/virtual-networks-overview

Austinmccollum. (n.d.). *Overview of log analytics in Azure Monitor - Azure Monitor*. Microsoft Learn. Retrieved April 12, 2025, from https://learn.microsoft.com/en-us/azure/azure-monitor/logs/log-analytics-overview?tabs=simple

Azevedo, P., & Santos, V. (2022). YOLO-Based object detection and tracking for autonomous vehicles using edge devices. In *Lecture notes in networks and systems* (pp. 297–308). https://doi.org/10.1007/978-3-031-21065-5_25

Bay, H., Tuytelaars, T., & Van Gool, L. (2006). SURF: Speeded up robust features. In *Lecture notes in computer science* (pp. 404–417). https://doi.org/10.1007/11744023_32

Benevides, A. (2022, January 6). Introduction to MQTT and configuration of a Mosquitto Broker. *Medium*. https://andre-benevides.medium.com/introduction-to-mqtt-and-configuration-of-a-mosquitto-broker-f0f7a7738bc8

Bliccathemes. (n.d.-a). *HTS*. Retrieved April 11, 2025, from http://www.htsol.com/

Bliccathemes. (n.d.-b). *Imaging Units | HTS*. Retrieved April 11, 2025, from http://www.htsol.com/products/imaging-units/

Bliccathemes. (n.d.-c). *Lane Controllers | HTS*. Retrieved April 11, 2025, from http://www.htsol.com/products/lane-controllers/

Boesch, G. (2025, January 31). *YOLO Explained: From v1 to v11*. viso.ai. Retrieved April 11, 2025, from https://viso.ai/computer-vision/yolo-explained/

Breck, E., Cai, S., Nielsen, E., Salib, M., Sculley, D., & Google, Inc. (2017). The ML Test score: a rubric for ML production readiness and technical debt reduction. In *IEEE* [Journal-article]. https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/aad9f93b86b7add-fea4c419b9100c6cdd26cacea.pdf

Buslaev, A., Iglovikov, V. I., Khvedchenya, E., Parinov, A., Druzhinin, M., & Kalinin, A. A. (2020). Albumentations: fast and flexible image augmentations. *Information*, *11*(2), 125. https://doi.org/10.3390/info11020125

Bwren. (n.d.). *Azure Monitor overview - Azure Monitor*. Microsoft Learn. Retrieved April 12, 2025, from https://learn.microsoft.com/en-us/azure/azure-monitor/fundamentals/overview

Carion, N., Massa, F., Synnaeve, G., Kirillov, A., & Zagoruyko, S. (2020, May 26). *End-to-End Object Detection with Transformers*. arXiv.org. Retrieved April 11, 2025, from https://arxiv.org/abs/2005.12872v3

Chen, Y., Wang, S., Lin, L., Cui, Z., & Zong, Y. (2024). Computer vision and deep learning transforming image recognition and beyond. *International Journal of Computer Science and Information Technology*, *2*(1), 45–51. https://doi.org/10.62051/ijcsit.v2n1.06

ColorVu. (n.d.). Hikvision. Retrieved April 11, 2025, from https://www.hikvision.com/europe/core-technologies/see-clearer-technology/colorvu/

Dahua Technology - world leading Video-Centric AIOT solution & service provider. (n.d.). Dahua Technology. Retrieved March 10, 2024, from https://www.dahuasecurity.com/

Data Version Control · DVC. (n.d.). Data Version Control · DVC. Retrieved April 14, 2025, from https://dvc.org/

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L., & Dept. of Computer Science, Princeton University, USA. (n.d.). ImageNet: a Large-Scale hierarchical image database. *Dept. Of Computer Science, Princeton University, USA*. https://www.image-net.org/static_files/papers/imagenet_cvpr09.pdf

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., & Houlsby, N. (2020, October 22). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv.org. Retrieved April 11, 2025, from https://arxiv.org/abs/2010.11929

Dotnet. (n.d.). *GitHub - dotnet/MQTTnet: MQTTnet is a high performance .NET library for MQTT based communication. It provides a MQTT client and a MQTT server (broker). The implementation is based on the documentation from http://mqtt.org/.* GitHub. Retrieved April 11, 2025, from https://github.com/dotnet/MQTTnet

DS-2XS6A46G1-IZS/C36S80. (n.d.). Hikvision. Retrieved April 9, 2025, from https://www.hikvision.com/europe/products/IP-Products/Network-Cameras/solar-powered-security-camera-setup/ds-2xs6a46g1-izs-c36s80/

Edwards, S., Kubernetes Security WG, Kubernetes, & Trail of Bits. (2019). *Kubernetes Security Whitepaper* [Press release]. Trail of Bits Kubernetes Assessment White Paper.

Elharrouss, O., Akbari, Y., Almadeed, N., & Al-Maadeed, S. (2024). Backbones-review: Feature extractor networks for deep learning and deep reinforcement learning approaches in computer vision. *Computer Science Review*, *53*, 100645. https://doi.org/10.1016/j.cosrev.2024.100645

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2009). The Pascal Visual Object Classes (VOC) challenge. *International Journal of Computer Vision*, *88*(2), 303–338. https://doi.org/10.1007/s11263-009-0275-4

Faster RCNN vs Retinanet Comparison | Restackio. (n.d.). Retrieved April 14, 2025, from https://www.restack.io/p/computer-vision-answer-faster-rcnn-vs-retinanet-cat-ai

Garikapati, D., & Shetiya, S. S. (2024). Autonomous Vehicles: Evolution of artificial intelligence and the current industry landscape. *Big Data and Cognitive Computing*, *8*(4), 42. https://doi.org/10.3390/bdcc8040042

General Data Protection Regulation (GDPR) – legal text. (2024, April 22). General Data Protection Regulation (GDPR). https://gdpr-info.eu/

Ggailey. (n.d.). *Azure Functions overview*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/azure-functions/functions-overview

Girshick, R. & Microsoft Research. (n.d.). Fast R-CNN. In *Fast R-CNN* [Journal-article]. https://openaccess.thecvf.com/content_iccv_2015/papers/Girshick_Fast_R-CNN_ICCV_2015_paper.pdf

GitHub. (2024, July 29). *What is a DevOps pipeline? A complete guide*. GitHub. https://github.com/resources/articles/devops/pipeline

Halkazwini. (n.d.). *Azure Network Watcher overview*. Microsoft Learn. Retrieved April 12, 2025, from https://learn.microsoft.com/en-us/azure/network-watcher/network-watcher-overview

He, K., J., Gkioxari, G., Doll´Ar, P., & Facebook AI Research (FAIR). (2018). Mask R-CNN. In *arXiv* [Report]. https://arxiv.org/abs/1703.06870v3

Hikvision Global - Leading the future of AIoT. (n.d.). Hikvision. Retrieved April 14, 2025, from https://www.hikvision.com/en/

HTS. (n.d.). *HTS Vehicle Recognition Solutions*. http://www.htsol.com/wp-content/uploads/2018/01/N70_Data_Sheet.pdf

HTS Vehicle Recognition Solutions. (n.d.). *HTS Vehicle Recognition Solutions*. http://www.htsol.com/wp-content/uploads/2014/08/SeeControl_VRS_small.pdf

HTS Vehicle Recognition Solutions. (2021). *HTS Vehicle Recognition Solutions LC4100 Series Rack mount lane Controller and VRS Server* [Technical Manual; Hardware]. HTS. http://www.htsol.com/wp-content/uploads/2018/02/VRS_Lane_Controller_LC4100_Series_DataSheet.pdf

Hui, J. (2018, March 28). Object detection: speed and accuracy comparison (Faster R-CNN, R-FCN, SSD, FPN, RetinaNet and YOLOv3). *https://jonathan-hui.medium.com*. https://jonathan-hui.medium.com/object-detection-speed-and-accuracy-comparison-faster-r-cnn-r-fcn-ssd-and-yolo-5425656ae359

Hui, J. (2020, February 6). mAP (mean Average Precision) for Object Detection - Jonathan Hui - Medium. *Medium*. https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

HumanSignal. (n.d.). *GitHub - HumanSignal/label-studio: Label Studio is a multi-type data labeling and annotation tool with standardized output format*. GitHub. https://github.com/HumanSignal/label-studio

IP ratings. (n.d.). International Electrotechnical Commission. Retrieved April 10, 2025, from https://www.iec.ch/ip-ratings

Jana, A. P., Biswas, A., & Mohana, N. (2018). YOLO based Detection and Classification of Objects in video records. *ROBOT2022: Fifth Iberian Robotics Conference*. https://doi.org/10.1109/rteict42901.2018.9012375

Karypidis, E., Mouslech, S. G., Skoulariki, K., & Gazis, A. (2022). Comparison analysis of traditional machine learning and deep learning techniques for data and image classification. *WSEAS TRANSACTIONS ON MATHEMATICS*, *21*, 122–130. https://doi.org/10.37394/23206.2022.21.19

Kreuzberger, D., Kühl, N., & Hirschl, S. (2023). Machine Learning Operations (MLOPs): overview, definition, and architecture. *IEEE Access*, *11*, 31866–31879. https://doi.org/10.1109/access.2023.3262138

Li, M., Bi, Z., Wang, T., Wen, Y., Niu, Q., Liu, J., Peng, B., Zhang, S., Pan, X., Xu, J., Wang, J., Chen, K., Yin, C. H., Feng, P., & Liu, M. (2024). Deep Learning and Machine Learning with GPGPU and CUDA: Unlocking the Power of Parallel Computing. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2410.05686

Light, R. A. (2017). Mosquitto: server and client implementation of the MQTT protocol. *The Journal of Open Source Software*, *2*(13), 265. https://doi.org/10.21105/joss.00265

Lin, T., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017, August 7). *Focal loss for dense object detection*. arXiv.org. Retrieved April 11, 2025, from https://arxiv.org/abs/1708.02002v2

Lin, T., Maire, M., Belongie, S., Bourdev, L., Girshick, R., Hays, J., Perona, P., Ramanan, D., Zitnick, C. L., & Dollár, P. (2014, May 1). *Microsoft COCO: Common Objects in context*. arXiv.org. Retrieved April 11, 2025, from https://arxiv.org/abs/1405.0312

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In *Lecture notes in computer science* (pp. 21–37). https://doi.org/10.1007/978-3-319-46448-0_2

marcomweb. (n.d.). *Milesight SC211-EU 4G - Price*. https://shop.marcomweb.it. Retrieved April 11, 2025, from https://shop.marcomweb.it/en/shop-online/brand/milesight/milesight-camera/sc211-eu-4g-solar-powered-traffic-sensing-camera-dettagli.html?srsltid=AfmBOoq4f6JDmpS6518HV14bjk-bRfWbzhDr04TZ8R0ppFq0wJQTtGTK1

Milesight. (2022). *4G Solar-powered Traffic Sensing Camera - User Manual*. https://resource.milesight.com/milesight/security/document/user-manual/milesight-4g-solar-powered-traffic-sensing-camera-user-manual.pdf

MileSight. (2024, October 30). *Milesight | 5G, AI, IoT and LoRAWAN®*. Milesight. Retrieved April 9, 2025, from https://www.milesight.com/

Milesight: SC211-NA/EU/AU (2MP). (2024). 4g-solar-powered-traffic-sensing-camera. Retrieved April 10, 2025, from https://www.milesight.com/security/product/4g-solar-powered-traffic-sensing-camera

MLFlow Documentation. (n.d.). https://www.mlflow.org/docs/1.29.0/index.html

Ml-ops.org. (2025, March 24). Retrieved April 12, 2025, from https://ml-ops.org/

Modern app architecture for the enterprise. (n.d.). In *Modern Application Architecture for the Enterprise*.

Mohanty, S. P., Choppali, U., & Kougianos, E. (2016). Everything you wanted to know about smart cities: The Internet of things is the backbone. *IEEE Consumer Electronics Magazine*, *5*(3), 60–70. https://doi.org/10.1109/mce.2016.2556879

Neamah, S. B., & Karim, A. A. (2023). Real-time traffic monitoring system based on deep learning and YOLOV8. *ARO-The Scientific Journal of Koya University*, *11*(2), 137–150. https://doi.org/10.14500/aro.11327

netcamcenter. (2024). *HikVision DS-2XS6A46G1/P-IZS - Price*. Netcamcenter. https://netcamcenter.com/en/products/ip-cameras/ds-2xs6a46g1-p-izs-c36s802-8-12mm

Oguine, K. J., Oguine, O. C., & Bisallah, H. I. (2022). YOLO v3: Visual and Real-Time Object Detection Model for Smart Surveillance Systems [Journal-article]. *University of Abuja*. https://arxiv.org/abs/2209.12447v1

Omniq. (2024, July 30). *PERCS parking - Omniq*. Retrieved April 11, 2025, from https://omniq.com/smart-parking/permit-enforcement-revenue-collection-percs/

O'Shea, K., & Nash, R. (2015, November 26). *An introduction to convolutional neural networks*. arXiv.org. Retrieved April 11, 2025, from https://arxiv.org/abs/1511.08458

PCI Security Standards Council, LLC. (2018). *PCI DSS Quick Reference Guide*. https://listings.pcisecuritystandards.org/documents/PCI_DSS-QRG-v3_2_1.pdf

Pölöskei, I. (2021). MLOps approach in the cloud-native data pipeline design. *Acta Technica Jaurinensis*, *15*(1), 1–6. https://doi.org/10.14513/actatechjaur.00581

Premaratne, P., Kadhim, I. J., Blacklidge, R., & Lee, M. (2023). Comprehensive review on vehicle Detection, classification and counting on highways. *Neurocomputing*, *556*, 126627. https://doi.org/10.1016/j.neucom.2023.126627

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015, June 8). *You only look once: Unified, Real-Time Object Detection*. arXiv.org. Retrieved April 11, 2025, from https://arxiv.org/abs/1506.02640v5

Reis, D. H. D., Welfer, D., De Souza Leite Cuadros, M. A., & Gamarra, D. F. T. (2019). Mobile Robot Navigation Using an Object Recognition Software with RGBD Images and the YOLO Algorithm. *Applied Artificial Intelligence*, *33*(14), 1290–1305. https://doi.org/10.1080/08839514.2019.1684778

RekOr - Delivering revolutionary roadway Intelligence. (n.d.). Retrieved April 11, 2025, from https://www.rekor.ai/

Ren, S., He, K., Girshick, R., & Microsoft Research. (n.d.). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In *Microsoft Research* [Journal-article]. https://proceedings.neurips.cc/paper_files/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf

Rights, O. F. C. (2025, March 14). *Summary of the HIPAA Privacy Rule*. HHS.gov. https://www.hhs.gov/hipaa/for-professionals/privacy/laws-regulations/index.html

S, R. B. (2020). Object Detection using Region based Convolutional Neural Network: A Survey. *International Journal for Research in Applied Science and Engineering Technology*, *8*(7), 1927–1932. https://doi.org/10.22214/ijraset.2020.30430

Schmidhuber, J. (2014). Deep learning in neural networks: An overview. *Neural Networks*, *61*, 85–117. https://doi.org/10.1016/j.neunet.2014.09.003

SD2A400HB-GN-AGQ-PV-SP-EAU. (n.d.). Dahua Technology. Retrieved April 11, 2025, from https://www.dahuasecurity.com/mena/products/All-Products/Discontinued-Products/PTZ-Cameras/SD2A400HB-GN-AGQ-PV-SP-EAU

Sekhar, C., & Meghana, P. S. (2020). A study on backpropagation in artificial neural networks. *Asia-Pacific Journal of Neural Networks and Its Applications*, *4*(1), 21–28. https://doi.org/10.21742/ajnnia.2020.4.1.03

shopdelta. (n.d.). *DHAUA SD2A400HB-GN-AGQ-PV-SP-EA- Price*. Shopdelta. https://shopdelta.eu/solar-ip-camera-outdoor-sd2a400hb-gn-agq-pv-sp-eau-pir-4glte-3-7-mpx-4-mm-dahua_l2_p20705.html

Singh, R. (2024, November 15). Understanding and implementing faster R-CNN - Rishabh Singh - medium. *Medium*. https://medium.com/@RobuRishabh/understanding-and-implementing-faster-r-cnn-248f7b25ff96

Sojasingarayar, A. (2024, February 23). Faster R-CNN vs YOLO vs SSD — Object Detection Algorithms. *Medium*. https://medium.com/ibm-data-ai/faster-r-cnn-vs-yolo-vs-ssd-object-detection-algorithms-18badb0e02dc

Subramanya, R., Sierla, S., & Vyatkin, V. (2022). From DevOps to MLOps: Overview and application to Electricity Market Forecasting. *Applied Sciences (Switzerland)*. https://doi.org/10.3390/app12199851

Terayama, M., Shin, J., Chang, W.-D., & Graduate School of Computer Science and Engineering, The University of Aizu, Tsuruga, Ikki-machi Aizuwakamatsu City, Fukushima, 965-8580, Japan. (2009). Object Detection using Histogram of Oriented Gradients. In *Conference Paper*.

The center for all your data, analytics, and AI – Amazon SageMaker – AWS. (n.d.). Amazon Web Services, Inc. https://aws.amazon.com/sagemaker/

Tomvcassidy. (n.d.). *Azure Container Instances documentation - serverless containers, on demand*. Microsoft Learn. https://learn.microsoft.com/en-us/azure/container-instances/

Usmani, M. F. & Frankfurt University of Applied Sciences. (2021). MQTT Protocol for the IoT - Review Paper. In *Technical Report* [Report]. https://doi.org/10.13140/RG.2.2.26065.10088

Wang, R., Xia, Y., Wang, G., & Tian, J. (2015). License plate localization in complex scenes based on oriented FAST and rotated BRIEF feature. *Journal of Electronic Imaging*, *24*(5), 053011. https://doi.org/10.1117/1.jei.24.5.053011

Wazir, S., Kashyap, G. S., & Saxena, P. (2023). MLOps: A review. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.2308.10908

Weihong, W., & Jiaoyang, T. (2020). Research on license plate recognition algorithms based on deep learning in complex environment. *IEEE Access*, *8*, 91661–91675. https://doi.org/10.1109/access.2020.2994287

Yin, W., Kann, K., Yu, M., & Schütze, H. (2017). Comparative study of CNN and RNN for natural language processing. *arXiv (Cornell University)*. https://doi.org/10.48550/arxiv.1702.01923

YOLOV8: State-of-the-Art Computer Vision Model. (n.d.). https://yolov8.com/

APPENDICES

PERCS OVERVIEW

All users of PERCS access to the same PERCS portal. Role based claims manage user's accessibility. It has a well-customized dynamic role-claim implementation, and it can be created by any user who has authorization to access role section.

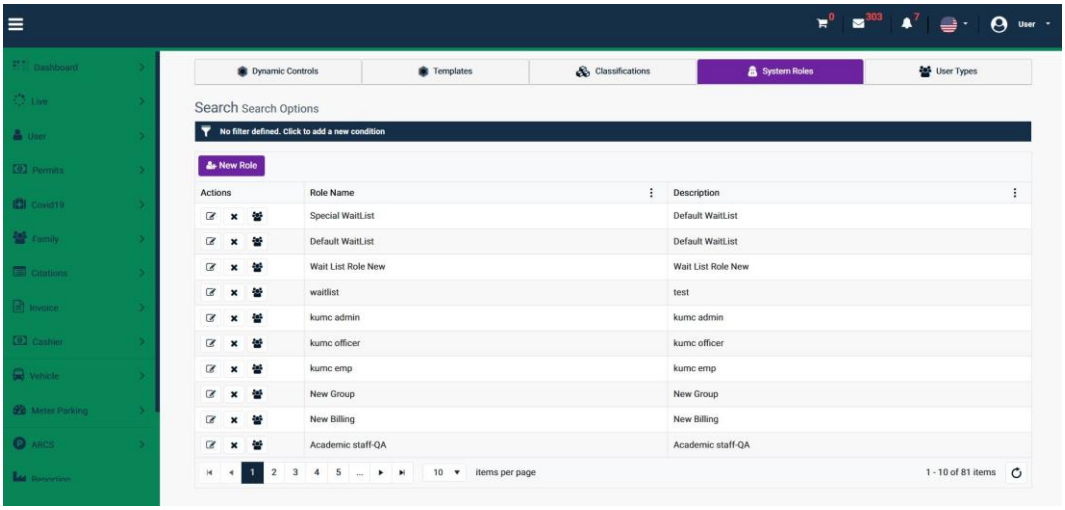


Figure 23. System Roles

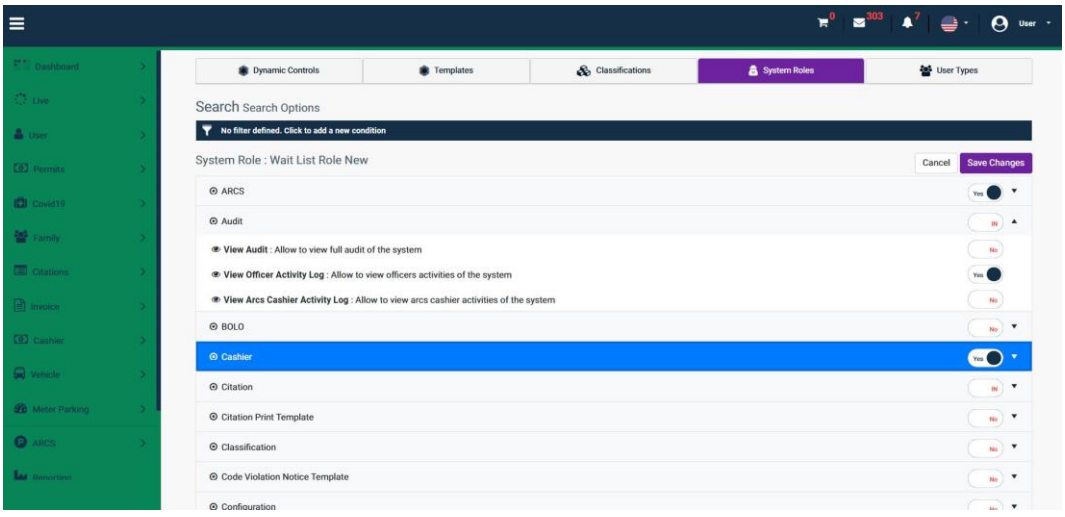


Figure 24. Create/Update Role

Users

All users of the system can view in Users Grid. It can be operated by authorized members.

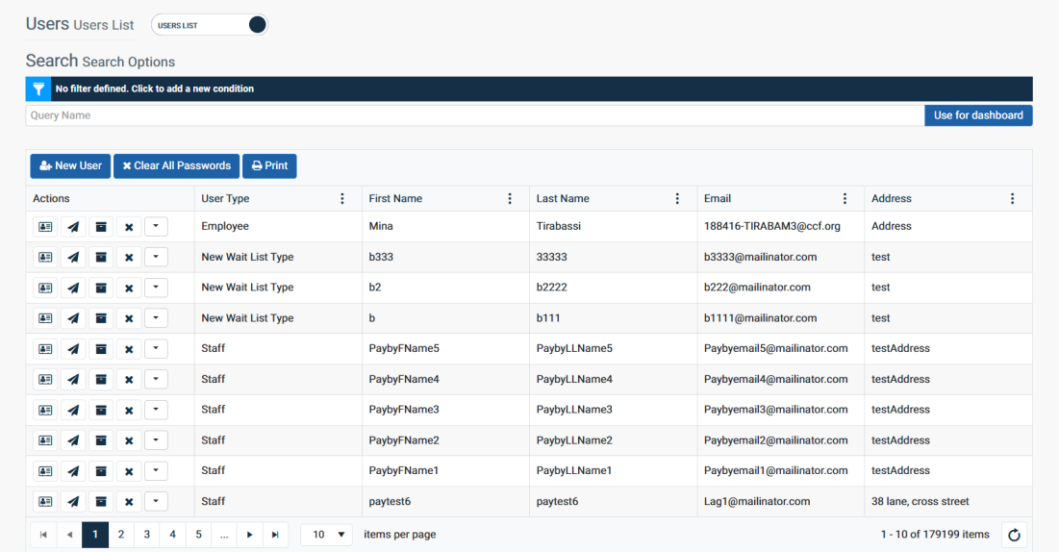


Figure 25. User List

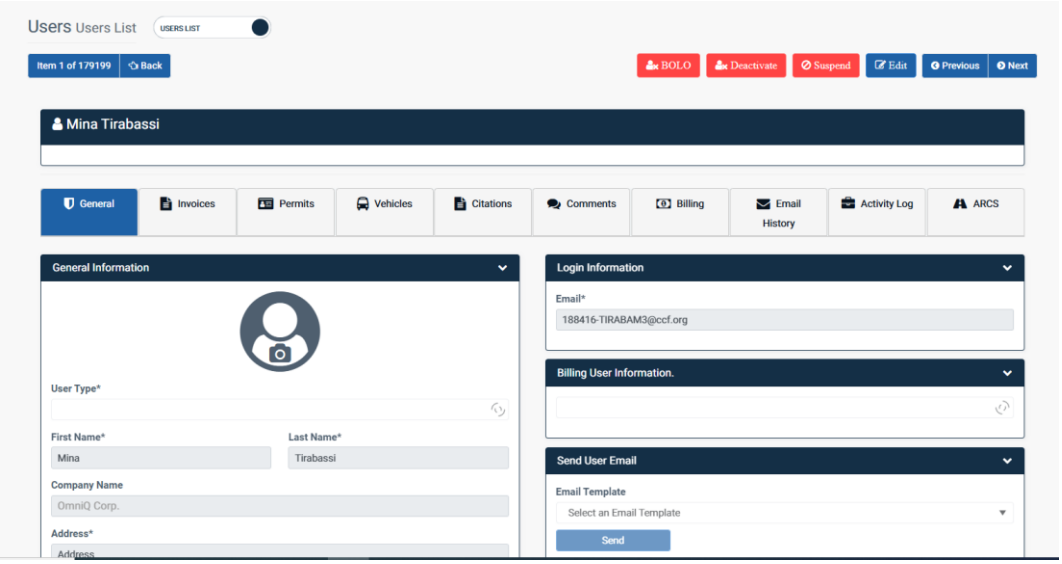


Figure 26. User Profile View

Basically, PERCS have three major entities Permits, Citations and Vehicles. All the features and functionalities are implemented with these three entities for either known or unknown users.

Permits

Users can purchase permits themselves. Also, administrators can create permits instead of users.

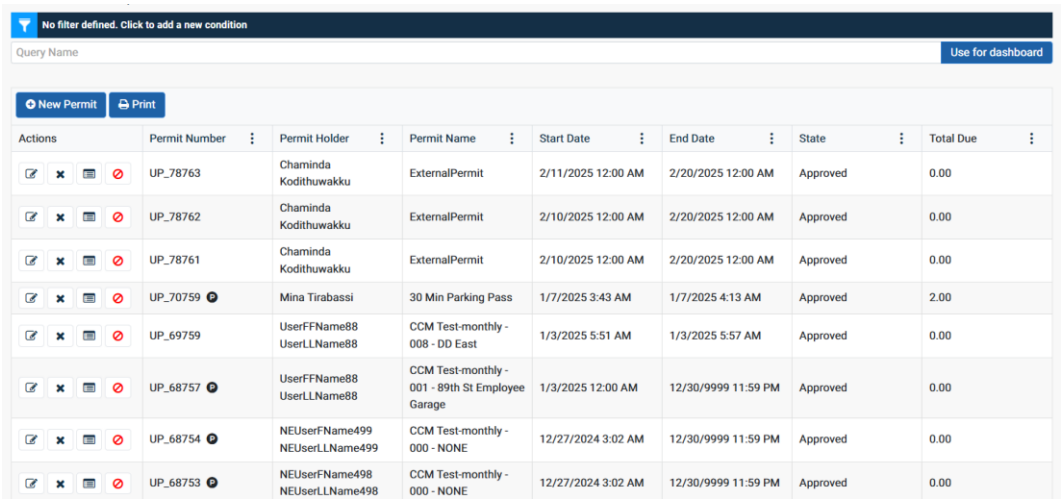


Figure 27. Permit List

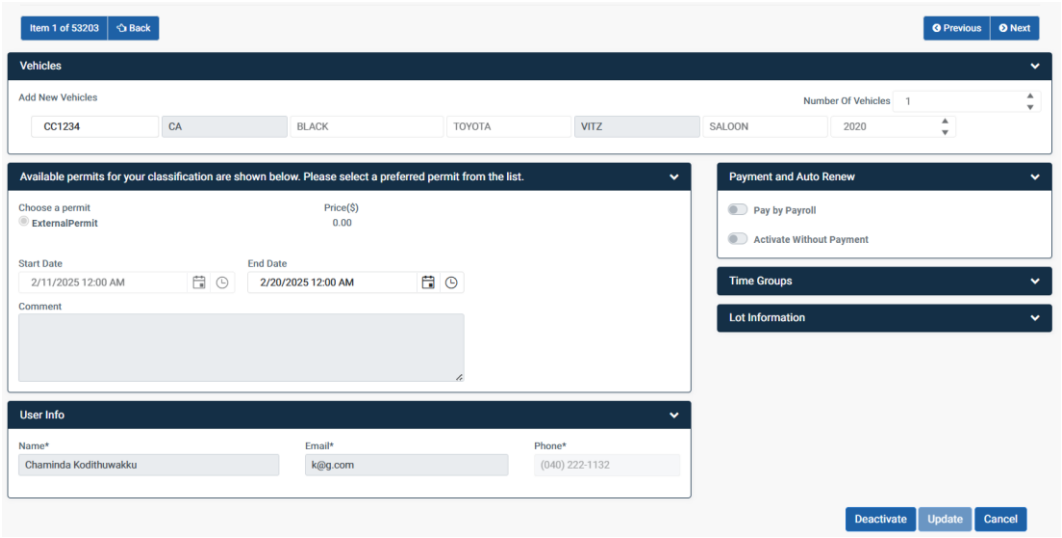


Figure 28. Permit View

Citations

Citations are fines that are issued by parking officers for parking violations. Parking officers use handheld devices to issue citation tickets and PERCS system update at the same time through the relevant REST APIs.

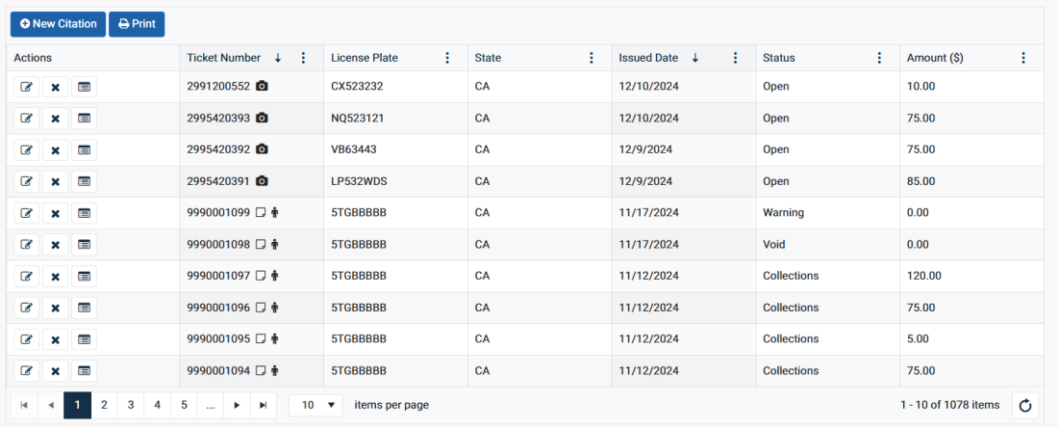


Figure 29. Citation List

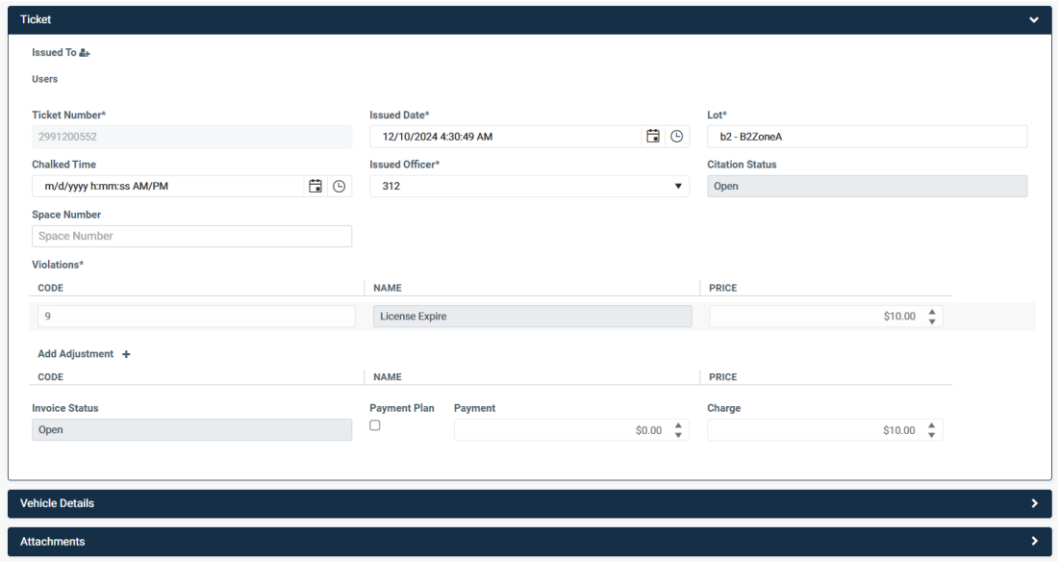


Figure 30. Citation View

Dashboard

All users can see a personalized dashboard whenever user login to the system. also, administrators or authorized users can customize the dashboard with adding different dashboard widgets.

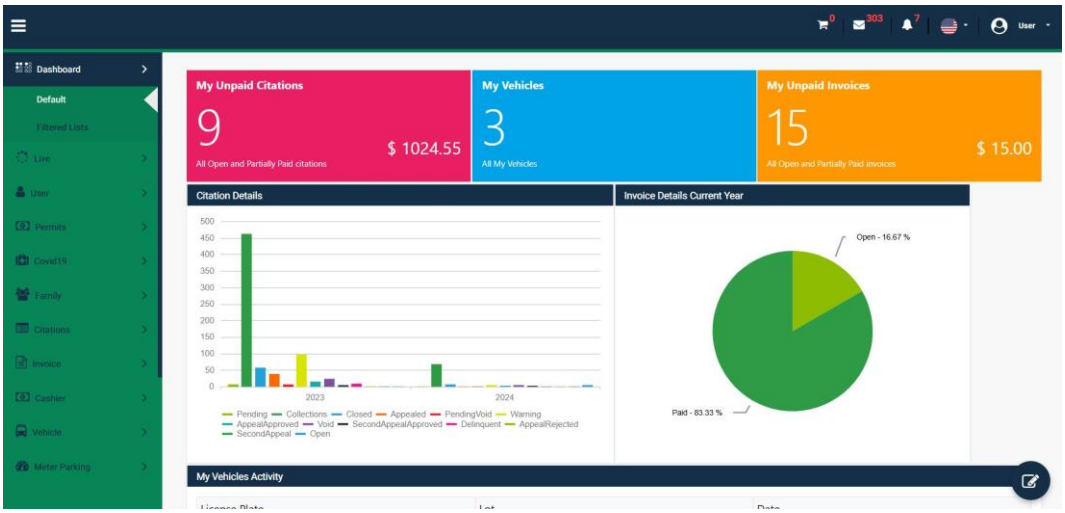


Figure 31. User Dashboard

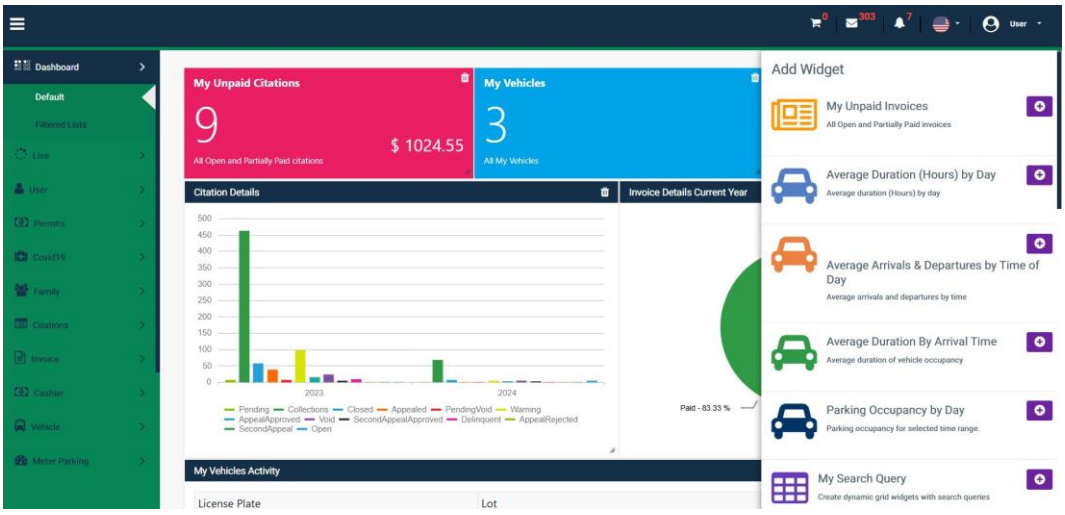
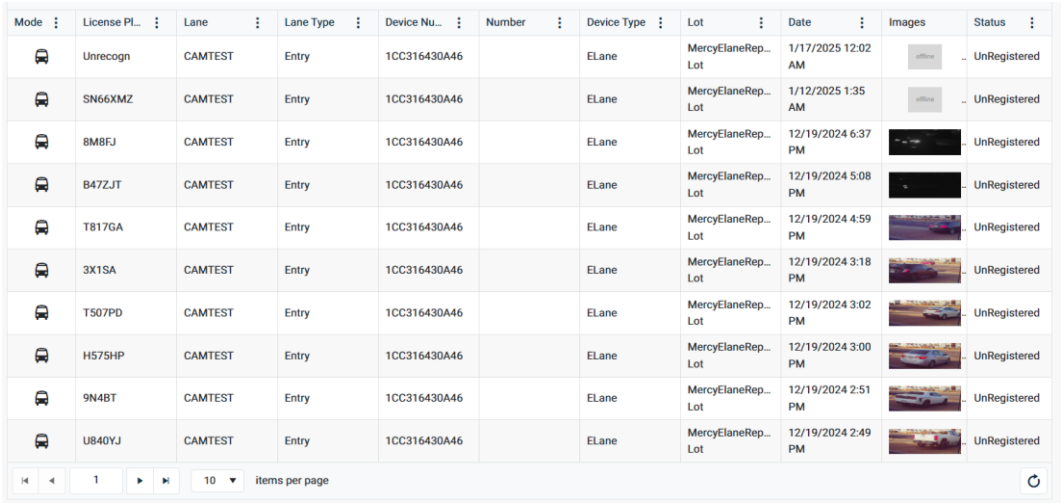


Figure 32. Dashboard Customization

Also, the most crucial part of the system is vehicles and vehicle records when they arrive and depart from parking locations. All information captures from surveillance cameras and feed to the system. and they record in vehicle inventory view. It records all details of vehicles and use them to handheld devices and mobile applications to track parking violations.



Mode	License Pl...	Lane	Lane Type	Device Nu...	Number	Device Type	Lot	Date	Images	Status
	Unrecogn	CAMTEST	Entry	1CC316430A46		ELane	MercyElaneRep... Lot	1/17/2025 12:02 AM		UnRegistered
	SN66XMZ	CAMTEST	Entry	1CC316430A46		ELane	MercyElaneRep... Lot	1/12/2025 1:35 AM		UnRegistered
	8M8FJ	CAMTEST	Entry	1CC316430A46		ELane	MercyElaneRep... Lot	12/19/2024 6:37 PM		UnRegistered
	B47ZJT	CAMTEST	Entry	1CC316430A46		ELane	MercyElaneRep... Lot	12/19/2024 5:08 PM		UnRegistered
	T817GA	CAMTEST	Entry	1CC316430A46		ELane	MercyElaneRep... Lot	12/19/2024 4:59 PM		UnRegistered
	3X1SA	CAMTEST	Entry	1CC316430A46		ELane	MercyElaneRep... Lot	12/19/2024 3:18 PM		UnRegistered
	T507PD	CAMTEST	Entry	1CC316430A46		ELane	MercyElaneRep... Lot	12/19/2024 3:02 PM		UnRegistered
	H575HP	CAMTEST	Entry	1CC316430A46		ELane	MercyElaneRep... Lot	12/19/2024 3:00 PM		UnRegistered
	9N4BT	CAMTEST	Entry	1CC316430A46		ELane	MercyElaneRep... Lot	12/19/2024 2:51 PM		UnRegistered
	U840YJ	CAMTEST	Entry	1CC316430A46		ELane	MercyElaneRep... Lot	12/19/2024 2:49 PM		UnRegistered

1 10 items per page

Figure 33. Vehicle Inventory.

Vehicle Detection Model

https://github.com/kodithuwa/VehicleDetectModel.git