TagT: A Versatile ANPR Solution for Diverse Applications

BY

TAN JO FANG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2025

COPYRIGHT STATEMENT

This Final Year Project report is submitted in partial fulfillment of the requirements for the degree of Bachelor of Computer Science (Honours) at Universiti Tunku Abdul Rahman (UTAR). This Final Year Project report represents the work of the author, except where due acknowledgment has been made in the text. No part of this Final Year Project report may be reproduced, stored, or transmitted in any form or by any means, whether electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the author or UTAR, in accordance with UTAR's Intellectual Property Policy.

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to all those who have supported and guided me throughout the course of this project.

First and foremost, I wish to extend my profound thanks to my former supervisor, Ts. Dr. Ooi Boon Yaik, for his invaluable guidance, patient mentorship and insightful feedback. His expertise in Computer Science and constant encouragement were instrumental in shaping the direction of my research and navigating the challenges along the way. His mentorship was crucial in guiding this project towards a successful publication at the 2025 IEEE 11th International Conference on Smart Instrumentation, Measurement and Applications (ICSIMA).

Next, I would like to express my sincere thanks to my current supervisor, Ts. Tan Teik Boon for his continued support and constructive advice. His perspective was especially valuable during the critical stages of the project and helped me to strengthen the overall quality of this work.

Furthermore, I would like to extend my appreciation to my family and friends for their unwavering encouragement, patience and emotional support throughout this journey. Their belief in me has been a constant source of strength and motivation, especially during challenging times.

To all who have contributed to the success of this project, directly or indirectly—thank you very much.

ABSTRACT

Traditional Automatic Number Plate Recognition (ANPR) systems, which focus solely on license plate numbers detection and recognition are vulnerable to fraud. This project presents the design and implementation of TagT, an advanced ANPR framework that enhances security through multi-attribute car recognition. TagT integrates three key components: a YOLO11n model for high-speed car detection, a ResNet18 model with cosine similarity for intelligent frame optimization and the Gemini model for robust recognition of a car's license plate number, brand and colour. An extensive preliminary investigation justifies the selection of these models over numerous alternatives. The final, implemented system features a native iOS application and a Python back-end. A comprehensive evaluation was conducted to validate the prototype's performance, focusing on two key areas: **Efficiency** and **Accuracy**. The evaluation of the architecture's efficiency demonstrated a **92.4% reduction in frames** sent for analysis, which resulted in a **91.9% decrease in API costs** and an **87.6% decrease in API latency** compared to a baseline approach. Furthermore, the system's real-world accuracy was validated across 160 demanding tests in varied conditions, achieving an **average Overall Success Rate of 83.75%** and a near-perfect Car Brand Accuracy of 98.13%. Overall, TagT provides a versatile, cost-effective and scalable solution that successfully addresses the limitations of traditional ANPR, enhancing public safety and car management.

Area of Study - IoT Solution, Computer Vision

Keywords - Automatic Number Plate Recognition (ANPR), YOLO, Gemini, License Plate Number Recognition, Car Brand Recognition, Car Colour Recognition

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ANPR	Automatic Number Plate Recognition
IoT	Internet of Things
AI	Artificial Intelligence
YOLO	You Only Look Once
OCR	Optical Character Recognition
CNN	Convolutional Neural Networks
RON95	Research Octane Number 95
API	Application Programming Interface
HSV	Hue, Saturation, and Value
YOLO	You Only Look Once
ResNet	Residual Network
FMT	Free Malaysia Today
ULEZ	Ultra Low Emission Zone
CPMMS	Car Parking Monitoring and Management System
RAM	Random Access Memory
ALPR	Automatic License Plate Recognition
V-Net	Volumetric Network
C-Net	Convolutional Network
R-Net	Recurrent Network
RGB	Red, Green, Blue
SSD	Single Shot Detector
VCR	Visual Car Recognition
DPM	Deformable Part Model
SCDPL	Spatially Coherent Discriminative Pattern Learning
MIL	Multiple Instance Learning
HOG	Histogram of Oriented Gradients
LLC+SPM	Local-constraint Linear Coding with Spatial Pyramid Matching
BoF	Bag of Features
UMPSA	Universiti Malaysia Pahang Al-Sultan Abdullah
VIP	Visually Impaired Persons

OpenCV	Open-Source Computer Vision Library
SAM	Segment Anything Model
CBIR	Content-Based Image Retrieval
RAD	Rapid Application Development

CHAPTER 1 INTRODUCTION

This chapter provides the foundational context for the TagT project. It begins by defining the core problem of license plate fraud and the motivation for developing an advanced ANPR system. Following this, the chapter outlines the formal project objectives, the scope and direction of the work, and the key contributions of the project. Finally, it presents the organization of the subsequent chapters in this report.

1.1 Problem Statement

The rising incidence of license plate fraud poses a serious threat to public safety and operational security. Cars with fake or swapped license plates, often termed "ghost cars," are frequently linked to criminal activities such as theft and car cloning, as they allow perpetrators to evade detection by standard law enforcement. While traditional ANPR systems can only read license plate numbers, they lack the sophistication to detect this type of fraud since they cannot verify if a license plate legitimately belongs to the car it is attached to.

Furthermore, the rigidity of traditional, hardware-based ANPR systems presents a significant issue. These systems are often deployed as fixed, specialized units that are difficult and expensive to modify and upgrade. This inflexibility prevents them from incorporating advanced recognition features, such as car brand and colour identification, which are necessary to address modern security demands. This limitation creates a critical gap in the market for a flexible, software-based solution that can provide comprehensive vehicle identification.

1.2 Motivation

The primary motivation for this project is to address the growing threat of license plate fraud. A prominent real-world example in Malaysia involves the use of swapped license plates to illegally access subsidized RON95 petrol. Traditional ANPR systems are ineffective against such schemes. The development of TagT is driven by the urgent need to create an ANPR solution that can cross-reference multiple car attributes, which are license plate number, car brand and colour, to authenticate a car's identity, significantly strengthening public safety and regulatory enforcement.

A second motivation is to overcome the limitations of fixed, hardware-based ANPR systems. By developing a flexible, software-based solution that can run on a mobile device, this project aims to create a more accessible, scalable and easily updatable system. This would enable advanced security features, such as verifying authorized cars in a kindergarten environment, without the need for expensive and proprietary hardware.

1.3 Project Objectives

The primary objective of the TagT project is to overcome the limitations of traditional ANPR systems, particularly their vulnerability to license plate fraud and the inflexibility of hardware-based architectures. The project aims to develop an advanced, software-based solution that enhances security by accurately recognizing multiple car attributes, which are the license plate number, car brand and colour.

Moreover, the project is defined by critical performance objectives. The system must achieve real-time processing capabilities, operating with minimal latency. Concurrently, it must be cost-effective, incorporating an optimization strategy to minimize the operational expenses associated with advanced AI model usage, thereby ensuring the solution is financially sustainable.

Ultimately, the final objective is to deliver a robust and user-friendly prototype that validates the proposed solution. This involves demonstrating reliable performance across a range of varied, real-world environmental conditions and integrating IoT principles to ensure seamless connectivity, culminating in a functional proof-of-concept application.

1.4 Project Scope and Direction

The scope of the TagT project encompasses the complete design, development, implementation and evaluation of an advanced, software-based Automatic Number Plate Recognition (ANPR) system. The main deliverable is a functional proof-of-concept prototype that demonstrates the viability of a hybrid, car multi-attribute recognition architecture. The project is strictly defined as a software-based solution, intentionally avoiding proprietary hardware to ensure flexibility and cost-effectiveness. The technical scope is centred on the creation of a client-server system. This includes:

- **A native iOS front-end application**
  Developed in Swift, this mobile application will serve as the primary user interface. Its scope includes managing the device's camera, capturing a real-time video feed, transmitting data to the server and displaying the final, parsed results in a clear and intuitive manner.
- **A Python-based back-end server**
  This server will house the core AI logic. Its scope includes creating a web API to communicate with the client, processing incoming images and executing the multi-stage analysis pipeline.

The core direction of the project follows an evidence-based, comparative methodology. To ensure the final system is both optimal and justified, the project scope includes a comprehensive preliminary investigation into a wide range of state-of-the-art AI models. This investigation will systematically test and evaluate:

- Multiple object detection frameworks, including several YOLO variants, to select the most efficient and accurate car detector.
- A traditional modular pipeline approach, testing specialized models for license plate detection, Optical Character Recognition (OCR) license plate number recognition, car brand recognition, and car colour analysis.
- Several advanced, large-scale multimodal AI models including Gemini, Grok and atc, as a unified solution for car attribute recognition.
- Frame optimization techniques, such as using a ResNet18 model for feature extraction and similarity analysis, to reduce redundancy and minimize API costs.

The findings from this rigorous investigation will directly inform the final architectural design. The project will culminate in the delivery of a fully documented, functional prototype that has been validated against a series of real-world performance benchmarks.

1.5 Contributions

What makes this project's contribution particularly valuable is that it provides a comprehensive and transparent blueprint for the process of engineering a modern and high-performance computer vision system. The contribution is not just the final product, but the rigorous and data-driven methodology used to create and validate it. This report meticulously documents this journey, offering several key contributions to the field.

First, this work presents a detailed case study on the limitations of a traditional, modular ANPR pipeline. By quantitatively demonstrating the critical failure points of specialized models, particularly in Optical Character Recognition (OCR) reading and scalable car brands recognition, this report provides clear and empirical evidence for why a new architectural approach is necessary. This serves as a valuable lesson for developers, highlighting the hidden complexities and fragility of building a recognition system from multiple, disparate components.

Next, the project contributes a thorough, head-to-head comparative analysis of advanced AI models for a specific, real-world task. By testing these models under identical conditions and measuring their performance across multiple metrics, including accuracy, speed, cost and robustness to varying input resolutions, this work provides rare and valuable data that can inform the decisions of other developers and researchers when selecting a foundational model for their own applications.

Finally, the most significant contribution is a replicable and validated framework for an efficient hybrid ANPR architecture. The project proves that by using lightweight local models like YOLO and ResNet18 as an intelligent pre-filtering and optimization layer, it is possible to harness the power of a Gemini model in a way that is both financially sustainable and fast enough for real-time applications. The finding that this approach can reduce the volume of data sent for analysis by over 90%, with corresponding reductions in cost and latency, is a critical

contribution. It establishes a practical and effective design pattern for building the next generation of intelligent, real-time visual analysis systems.

1.6 Report Organizations

This report is organized into seven chapters, each structured to logically present the design methodology, implementation and evaluation of the TagT system.

- Chapter 1 provides the foundational context for the project, defining the problem statement and motivation and outlining the formal project objectives, scope and contributions.
- Chapter 2 presents a comprehensive literature review of relevant research, covering topics such as license plate fraud, the challenges of traditional ANPR systems and existing methodologies for vehicle, brand, and color recognition.
- Chapter 3 details the final system methodology and high-level design. It presents the key visual models of the finalized system, including the use case diagram, the system architecture diagram and the activity diagram that illustrates the operational workflow.
- Chapter 4 describes the extensive preliminary investigation and evidence-based design process that led to the final architecture. It details the comparative testing of various technologies, including a modular pipeline approach and a unified AI model approach and presents the experimental results that justify the selection of the final system components.
- Chapter 5 outlines the full implementation of the TagT system. It covers the hardware and software setups for both the front-end and back-end, details the specific system configurations, demonstrates the real-world operation of the final application with screenshots and discusses the challenges overcome during development.
- Chapter 6 presents a comprehensive evaluation of the implemented prototype. It details the performance metrics used and presents the results from two key tests: an architectural efficiency evaluation and a real-world accuracy assessment across various environmental conditions.
- Chapter 7 concludes the report by summarizing the project's key findings and outcomes. It also provides recommendations for potential future work and enhancements to the TagT system.

CHAPTER 2 LITERATURE REVIEW

In this chapter, we provide a thorough review of license plate fraud cases, current Automatic Number Plate Recognition (ANPR) systems and their challenges, car colour recognition, car brand recognition, object detection and real-time tracking, and image retrieval and similarity measurement. This review serves three main purposes: first, to understand the issues faced by only recognizing license plate numbers; second, to gain insights into the current ANPR systems and their challenges; and third, to explore methodologies for car colour recognition, car brand recognition, object detection and real-time tracking, and image retrieval and similarity measurement.

2.1 License Plate Fraud

License plate fraud, including cloning and swapping, presents significant challenges to car surveillance and makes multi-attribute recognition systems essential. FMT (Free Malaysia Today) Reporters [1] have reported instances of license plate swapping in Malaysia to exploit fuel subsidies, noting that a majority of detected cases resulted in legal penalties. Samuel [5] highlighted a case in United Kingdom where a cloned plate led to a misattributed fine, illustrating that a significant number of traffic fines are linked to cloning errors. McLogan [6] documented an increase in fake plates in New York where drivers bought vanity plates online and it is almost identical to real ones, help drivers avoid tolls and traffic rules. GB News [7] reported that a high percentage of Ultra Low Emission Zone (ULEZ) fines in London were related to plate scams, further emphasizing the prevalence of fraud. These real-world cases emphasize the need for TagT's comprehensive approach, which integrates license plate numbers, car brands and colours recognition to effectively detect fraudulent activities.

2.2 Automatic Number Plate Recognition (ANPR) and Its Challenges

Automatic Number Plate Recognition (ANPR) systems are important for car identification and it supports applications when in toll collection, parking management, and law enforcement. Aalsalem, Khan and Dhabbah [2] proposed an Automated Car Parking Monitoring and Management System (CPMMS) for Jazan University, employing ANPR cameras to capture license plate numbers at entrance/exit gates and parking lots. The system integrates a database to store vehicle and owner information, complemented by a mobile application that assists users in locating parked vehicles and reporting parking violations, such as vehicle damage or blockages, shown in Figure 2.2.1 [2].

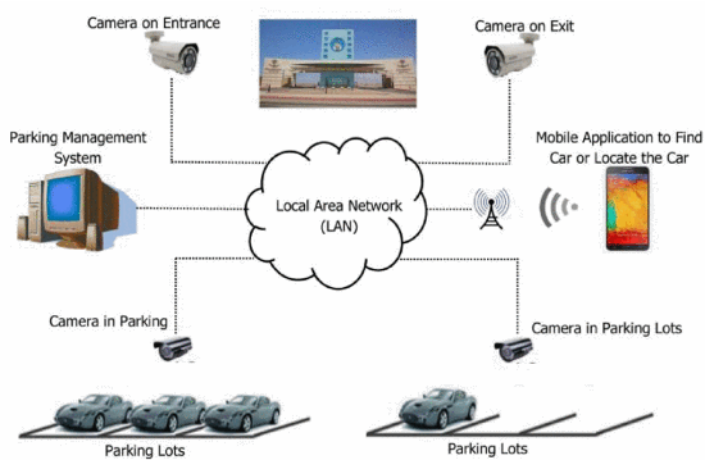


Figure 2.2.1 System Design by Aalsalem, Khan and Dhabbah [2]

Mustafa and Karabatak [3] conducted a systematic review of Automatic Number Plate Recognition (ANPR) systems, outlining key challenges impacting ANPR systems' accuracy and performance. These challenges are categorized into external and internal factors. External factors include plate variations such as plate size, plate position, plate colour, font style and so on, while environmental variations such as lighting conditions and surrounding effects, and camera mounting variations such as camera inclination and plate distance from camera. Internal factors encompass algorithmic limitations and hardware constraints such as camera shutter speed causing motion blur, resolution affecting image quality, focus length, view angle, and system RAM and processor specifications, as shown in Figure 2.2.2 [3].

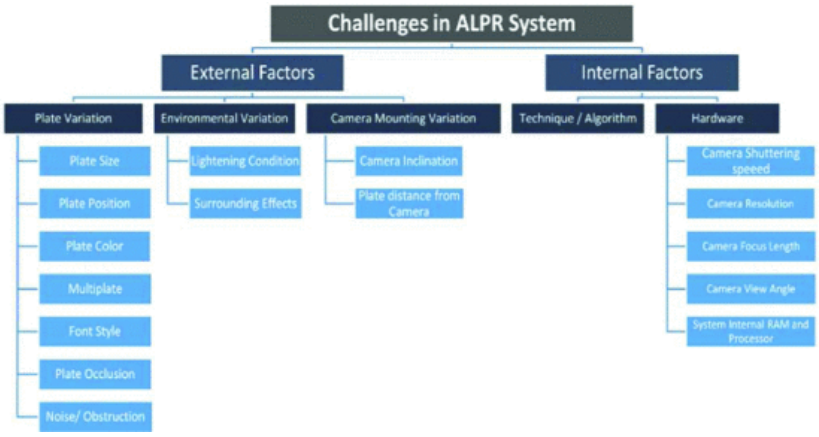


Figure 2.2.2 Challenges in ALPR System [3]

Marisekar et al. [4] developed a smart parking fare collection system by integrating ANPR with TensorFlow OCR to automate vehicle identification and billing. Their system has reduced billing time from 3 to 5 minutes in manual systems within 5 to 15 seconds, enhancing operational efficiency. They reported effective plate recognition across various conditions, with preprocessing techniques mitigating image blurring caused by adverse weather, such as heavy rain [4].

Kim, Kang, Kim and Yang [12] introduced an AI camera for on-device ANPR. They employed R-Net which is YOLOv2-based, V-Net which is ResNet blocks, and C-Net which is 18 convolutional layers, as shown in Figure 2.23. They have tested their system on 30,051 Korean plate images and achieved a 95% overall accuracy, with 99.89% license plate detection accuracy, 98.16% total recognition accuracy, and 93% character recognition accuracy [12].

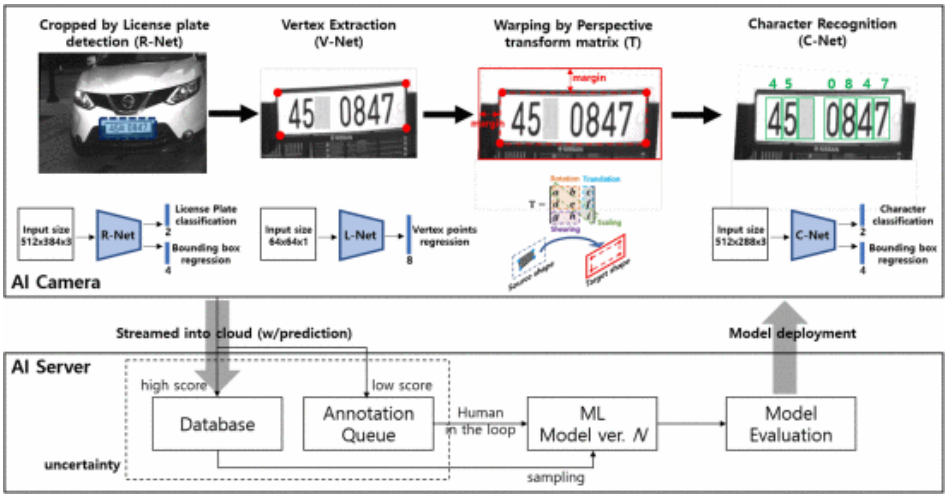


Figure 2.2.3 Methodology from Kim, Kang, Kim and Yang [12]

These studies demonstrate ANPR systems have potential to achieve high accuracies and yet environmental and regional factors still remain significant obstacles.

2.3 Car Colour Recognition

Car colour recognition is an essential complementary method for car identification, especially in scenarios where license plates are obscured, fraudulent, or missing. This section reviews three studies that developed different approaches to car colour recognition, detailing their methodologies and outcomes. Each study employs different techniques, ranging from colour space conversions and histogram analysis to deep learning-based feature extraction, to achieve reliable recognition in various environmental conditions.

Tong et al. [20] proposed a real-time vehicle colour recognition algorithm that combines RGB to HSV colour space conversion with sector-based histogram analysis and it is designed for embedded devices. The methodology involves capturing road videos using an IP camera, followed by background estimation to segment moving vehicles crossing a user-defined trip line [20]. A binary vehicle image T1 is generated via thresholding, and pixel values are extracted to compute the maximum (Max(R,G,B)) and minimum (Min(R,G,B)) in RGB colour space. An image T2 is derived using the formula T2 = Max(R,G,B) - Min(R,G,B), then segmented with an empirical threshold M1 = 128 to create a binary image T3 [20]. The pixel area ratio R = T1/T2 is calculated, and vehicles are classified into chromatic which has red, orange, yellow, green, cyan, blue, purple or achromatic which has black, white, grey categories using a threshold M2 = 0.3. For chromatic colours, the Hue (H) channel histogram in HSV space determines the colour based on the highest peak, while achromatic colours are identified by analysing histograms across five 72-degree sectors within a circular region centred on the vehicle's mass, with the majority vote determining the colour. They tested on 200 videos containing 795 vehicles, the algorithm achieved a 94.08% accuracy, correctly identifying 748 vehicles with 47 errors [20].

Agarwal, Shinde, Mohite and Jadhav [9] incorporated colour classification into a vehicle characteristic recognition framework using the YOLOv3 object detection model. The methodology uses YOLOv3 to detect vehicles in images and identify the bonnet area to select it as a prominent region for colour extraction [9]. A trained YOLOv3 model draws a bounding box around the bonnet and crops this region then extracts dominant colours by averaging the

RGB values from these colours. The averaged RGB value is compared to a database containing 28,907 sets of RGB values and corresponding colour names to determine the closest colour match [9]. They also used image enhancement techniques, such as brightening, sharpening, and smoothing to improve the performance for dark or nighttime images. The model was manually tested on thousands of images and has achieved nearly 95% accuracy as shown in Figure 2.3.1 [9].

Model	Accuracy
1. Vehicle Detection YOLO model	98-99%
2. Logo Detection YOLO model	97-99%
3. Make Detection Classification Model	93%
4. Color Detection	95%
5. Number Plate	92%

Figure 2.3.1 Agarwal, Shinde, Mohite and Jadhav [9]

Ghanem and Holliman [21] highlighted the impact of colour space on vehicle re-identification using a Siamese network with SSD Mobilenet V2, trained on the PRIMAVERA dataset of 636,246 side-view images of 13,963 vehicles captured in both daytime and nighttime conditions. RGB images were converted into multiple colour spaces such as RGB, HSV, YUV, LUV, nRGB, c1c2c3, 12-bit RGB, and n-bit grayscale [21]. The SSD Mobilenet V2 used 435,153 daytime and 27,315 nighttime images to train and used 76,203 daytime and 4,982 nighttime images for validation. On the outcome, YUV achieved the highest validation accuracy of 95.25% ± 0.41%, followed by 4-bit grayscale at 94.97% ± 0.42% and RGB at 94.65% ± 0.44% for mixed daytime and nighttime data as shown in Figure 2.3.2 [21].

Table 1. Accuracy of vehicle ID using both daytime and night-time data to train and validate the network.

	RGB	HSV	YUV	LUV	nRGB	c1c2c3	12-Bit RGB	8-Bit Red	8-Bit Gray	4-Bit Gray	2-Bit Gray	Binary
Training	95.32%	94.61%	**95.74%**	95.22%	92.80%	92.28%	95.56%	95.02%	95.10%	95.51%	93.72%	88.18%
Validation	94.65 ± 0.44%	93.75 ± 0.47%	**95.25 ± 0.41%**	94.51 ± 0.44%	91.95 ± 0.53%	90.05 ± 0.58%	94.62 ± 0.44%	93.87 ± 0.47%	94.67 ± 0.44%	94.97 ± 0.42%	92.78 ± 0.50%	88.45 ± 0.62%

Figure 2.3.2 Result from Ghanem and Holliman [21]

These studies show that vehicle colour recognition systems achieve accuracies around 95% with deep learning models and optimized colour spaces. However, challenges such as reflections, shadows, and low lighting, necessitating further advancements in preprocessing and feature extraction techniques.

2.4 Car Brand Recognition

Car brand recognition relies on unique visual features to enhance Automatic Number Plate Recognition (ANPR) in car identification systems. This section reviews three studies that developed methods for car brand recognition, providing an overview of their project, methodologies, and results.

Hu et al. [19] has developed an end-to-end real-time vehicle brand recognition system for surveillance videos, introducing the Visual Car Recognition (VCR) dataset to address challenges like intra-class variations and environmental noise. The methodology uses a Deformable Part Model (DPM) detector to identify cars in video frames, followed by Spatially Coherent Discriminative Pattern Learning (SCDPL) with Multiple Instance Learning (MIL) and Histogram of Oriented Gradients (HOG) features to learn discriminative patterns such as logos, grille shapes, and window corners with spatial coherence constraints [19]. When they tested 37,195 frontal-view images across 30 brands on the VCR dataset, the system achieved a 94.66% average per-class accuracy as shown in Figure 2.4.1. Their system also outperformed Local-constraint Linear Coding with Spatial Pyramid Matching (LLC+SPM), which is 85.75% accuracy, Convolutional Neural Networks (CNN), which is 70.62% accuracy, and Bag of Features (BoF), which is 58.15% accuracy as shown in Figure 2.4.1 [19].

Brand	Mazda	Chery	KIA	Nissan	Mitsubish	Skoda	SGMW	Hyundai	Chevrolet	Citroen	Average
LLC+SPM	0.87	0.87	0.72	0.78	**0.97**	**0.97**	0.97	0.63	0.86	0.94	0.8575
LLC	0.66	0.68	0.53	0.56	0.87	0.84	0.90	0.39	0.66	0.79	0.6881
BoF+SPM	0.74	0.75	0.59	0.64	0.94	0.93	0.90	0.47	0.72	0.84	0.7530
BoF	0.55	0.58	0.44	0.45	0.73	0.75	0.82	0.31	0.54	0.65	0.5815
CNN	0.61	0.60	0.59	0.55	0.62	0.89	0.77	0.39	0.76	0.77	0.7062
Disc. Patch	0.58	0.34	0.22	0.16	0.55	0.76	0.67	0.13	0.27	0.12	0.5469
MMDL	0.83	**0.97**	0.89	**0.97**	0.96	0.87	0.84	**0.96**	0.88	0.91	0.9136
Ours	**0.97**	0.90	**0.91**	0.91	0.90	**0.97**	**0.98**	0.92	**0.94**	**0.98**	**0.9466**

Figure 2.4.1 Result from Hu et al. [19]

Anuwa, Ramli and Zulkifli [10] aimed to develop a fast and accurate car logo recognition model for staff vehicles at Universiti Malaysia Pahang Al-Sultan Abdullah (UMPSA). They compared YOLOv8x and Microsoft Azure Custom Vision and recommended larger datasets and re-filtering could improve performance. They captured rear-view car images and pre-processed them by resizing, rotating, smoothing with Gaussian blur, and labelling logos with

bounding boxes [10]. YOLOv8x uses a single-stage neural network for quick logo detection, while Azure Custom Vision fine-tunes a pre-trained neural network for better accuracy. On the outcome, Azure Custom Vision is slightly better than YOLOv8x. In short, YOLOv8s is faster, ideal for real-time use, but struggled for small logos, where Azure Custom Vision is slower but more accurate and easier to use [10].

Agarwal, Shinde, Mohite and Jadhav [9] developed a vehicle recognition system using traffic camera images to identify car makes and logos, but they also face challenges with poor lighting and custom logos. They used YOLOv3 to detect vehicles and logos and used ResNet152v2 to classify car makes. On a dataset of 10,000 logo images, the system achieved 98%-99% accuracy for vehicle detection, 97%-99% for car logo detection, and 93% for car make classification as shown in Figure 2.3.1 [9].

These studies demonstrate the potential of deep learning models for vehicle brand and logo identification, achieving high accuracies in controlled conditions, although challenges still remain with small, non-standard, or obscured logos.

2.5 Object Detection and Tracking

Object detection and tracking are essential for real-time surveillance, allowing vehicle monitoring and detection of traffic violations. This section reviews four studies that used deep learning and computer vision techniques, presenting an overview of their projects, methodologies, and results.

A, R, Malini and Archana [16] aimed to enhance object detection for visually impaired persons (VIP) using live video and assisting them in identifying objects but there are challenges such as low-resolution images and cost constraints. They used YOLOv3 algorithm to detect objects in video frames by dividing images into grids and predicting bounding boxes and class probabilities in a single pass. They tested on the Microsoft COCO dataset with 500 images, the system achieved 94% accuracy and processed faster than methods like Single Shot Detector (SSD) and Faster R-CNN, proving its potential for assistive technologies [16].

Rahman, Ami and Ullah [14] aimed to develop a real-time system to detect wrong-way vehicles in Bangladesh to reduce accidents and traffic congestion by using traffic camera footage. They

used YOLOv3 to detect vehicles and create bounding boxes, followed by centroid tracking to monitor vehicles in a specific area. The direction was determined by comparing centroid heights across frames, which able to accurately identify the wrong-way vehicles [14]. Their project has tested on three 1280×720-pixel videos from Chittagong city, Bangladesh and the system achieved nearly 100% accuracy by correctly identifying all wrong-way vehicles. However, the system has minor errors due to the overlapping vehicles [14].

Islam and Horio [15] focused on developing a real-time system for face recognition, tracking, and counting people in Dubai mall videos. They also calculated their time within the frame in order to enhance security in public spaces. They used OpenCV to recognize faces by matching them to a stored database, then applied centroid tracking by assigning unique IDs and tracked individuals by calculating the distance between bounding box centres across frames, maintaining the same ID if the distance is small, as shown in Figure 2.5.1. The system also calculated time spent by each ID's presence. They have tested their system using shopping mall videos and the system successfully tracked people using centroid distances, although some IDs switched due to overlaps. Hence, the results have shown the potential for vehicle tracking [15].

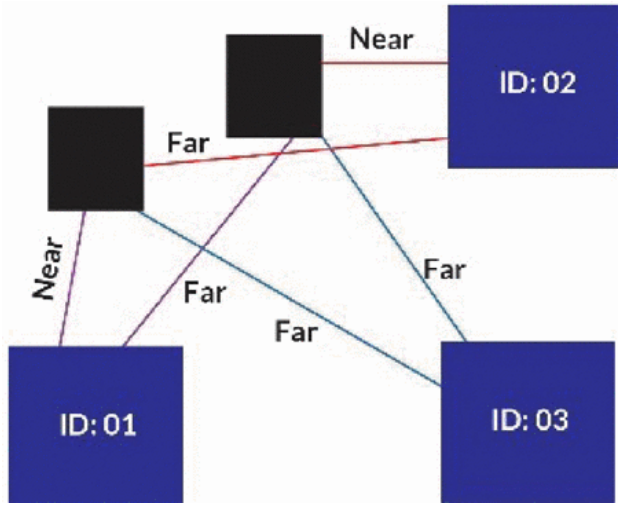


Figure 2.5.1 Methodology of Islam and Horio [15]

These studies show high accuracies in using YOLO for object detection and to track the object by using centroid tracking, but issues like occlusions and processing speed need further improvement for broader applications.

2.6 Image Retrieval and Similarity Measurement

Image retrieval and similarity measurement systems are essential by using similarity measurement to eliminate redundant frames and match query images against extensive databases, thereby achieving time and cost effectiveness. This section examines two studies, providing summaries of their projects, methodologies, and results.

Ozbek and Tekgoz [18] developed an image retrieval system for clothing and that could also be adapted to frame similarity verification. They used U2-Net to preprocess over 100,000 clothing images by removing backgrounds and categorizing them into upper body, lower body, and full body. After that, the authors applied ResNet-50 to extract embedding data for image comparison, while the Segment Anything Model (SAM) segmented user-uploaded query images, and K-Nearest Neighbours (K-NN) identified the five most similar images using Euclidean distance. On the outcome, the system achieved 92% accuracy on Euclidean similarity metric when tested on 100 products with 400 images, outperforming Cosine similarity metric is 80% accuracy and Manhattan similarity metric is 73% accuracy, as shown in Figure 2.6.1 [18].

Similarity metric	Accuracy
Cosine	80%
Manhattan	73%
Euclidean	**92%**

Figure 2.6.1 Result from Ozbek and Tekgoz [18]

Rani and Yuhandri [17] proposed a system to measure logo similarity for trademark verification in order to assist Indonesia's Ministry of Law and Human Rights in evaluating logo patent applications. They used the Content-Based Image Retrieval (CBIR) method to search a database of 210 logos and apply ResNet-18 to extract image features after data augmentation. The system was trained on 147 images, which is 70% from the database and validation on 63 images, which is 30% from the database with parameters of epoch=20, learning rate=0.00001, and mini-batch=5 to avoid overfitting [17]. After that, they have tested on four logos which are Sukamilktea, Exgen, Bete and Piniclean. The system achieved 93.65% accuracy after 84 iterations, with similarity scores of 82.80% from Sukamilktea, 100% from

Exgen, 96.36% from Bete, and 89.5% from Piniclean, as shown in Figure 2.6.2. Through the result, it shows the effectiveness for vehicle logo verification [17].

Logo Name	Test Logo	Image Similarity Results	Percentage of Similarity
Sukamilktea			82.80%
Exgen			100%
Bete			96.36%
Piniclean			89.5%

Figure 2.6.2 Result from Rani and Yuhandri [17]

CHAPTER 3 SYSTEM METHODOLOGY/APPROACH

This chapter details the high-level methodology and design of the TagT system. It serves as the architectural blueprint for the project, presenting a series of visual models that define the system's structure, user interactions and operational workflow. This chapter includes the system architecture diagram, the use case diagram with its description and the activity diagram.

3.1 Overview of the System

The proposed TagT system is designed to provide real-time identification of car attributes, including license plate number, car brand and car colour. For front-end, the system analyses a live video stream from a mobile device. For back-end, the system leverages a combination of object detection and a generative vision model to deliver fast and accurate results. By using a smartphone camera, the system offers a portable and cost-effective solution for automated car recognition.

The system architecture consists of three main components: a mobile front-end user interface, a back-end server and two integrated deep learning models. The user initiates the process by pointing their mobile camera at a car. The front-end application captures the video frames and transmits them to the back-end server for analysis.

The back-end first processes each frame using a YOLO11n-seg model to detect and segment any vehicles present. To optimize performance and ensure accuracy, the system incorporates a two-step quality filtering process. It first verifies that the detected car's area is significant enough and then checks that the image is not overly blurry by measuring its Laplacian variance against a set threshold. Frames that pass these checks are then sent to the Gemini model. This advanced vision model analyses the segmented car image to extract its license plate number, car brand and car colour. The processed information, along with performance metrics such as processing times and API costs, is then sent back to the front-end interface.

The mobile application displays these predicted outputs in real-time, allowing the user to instantly view the car's details. This streamlined, non-invasive approach provides a scalable and accessible tool for intelligent vehicle identification, suitable for various real-world applications.

3.2 System Architecture Diagram

This section presents the system architecture of the TagT system. The architecture outlines how the hardware and software components interact to detect and recognize car attributes, including the license plate number, car brand and car colour.

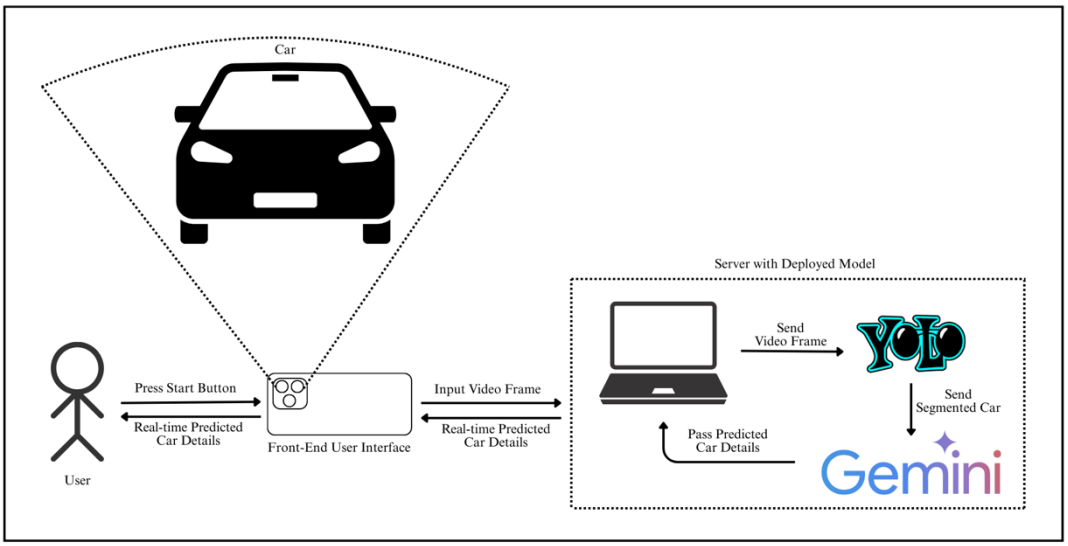


Figure 3.2.1 TagT System Architecture

As illustrated in the Figure 3.2.1, the system is composed of three primary components:

- A user
- A mobile phone serving as the front-end user interface
- A server with the deployed models

The process begins when the user initiates the system by pressing a start button on the mobile application. The phone's camera is activated and captures a continuous video stream of a car. This input video frame is then sent in real-time to the server for processing. The server is the core of the system and performs several critical steps:

1. **Car Detection and Segmentation**

   The incoming video frame is first processed by a YOLO11n-seg model. This model is responsible for detecting the presence of a car within the frame.

2. **Quality Filtering**

   This step ensures the detected car occupies a sufficient area of the frame and is not excessively blurry. If the frames fail one of these requirements, they will not proceed to the next step.

3. **Attribute Recognition**

   Once the frame passes the quality filters, the segmented image of the car is sent to the Gemini model. Gemini analyzes the image to extract specific details, which are the license plate number, car brand and colour.

4. **Data Transmission**

   Once the car details are generated, the server sends this information back to the front-end user interface.

The mobile application front-end user interface is developed using Xcode. It can receive and display the real-time predicted car details for the user. The information presented on the interface includes:

- License Plate Number
- Car Brand
- Car Colour
- Total Processing Time
- Gemini Processing Time
- Gemini API Cost

This architecture allows the user to access detailed vehicle information instantly through the mobile interface. By leveraging a mobile app for video capture and a powerful server for processing, the system provides a scalable and cost-effective solution for real-time automatic number-plate recognition (ANPR).

3.3 Activity Diagram

The section presents the activity diagram of the TagT system as shown in Figure 3.3.1. This diagram illustrates the sequential flow of operations carried out by the system, from the moment the user presses a button to open the camera. The system initiates by capturing a video frame, which is then sent to the server for processing.

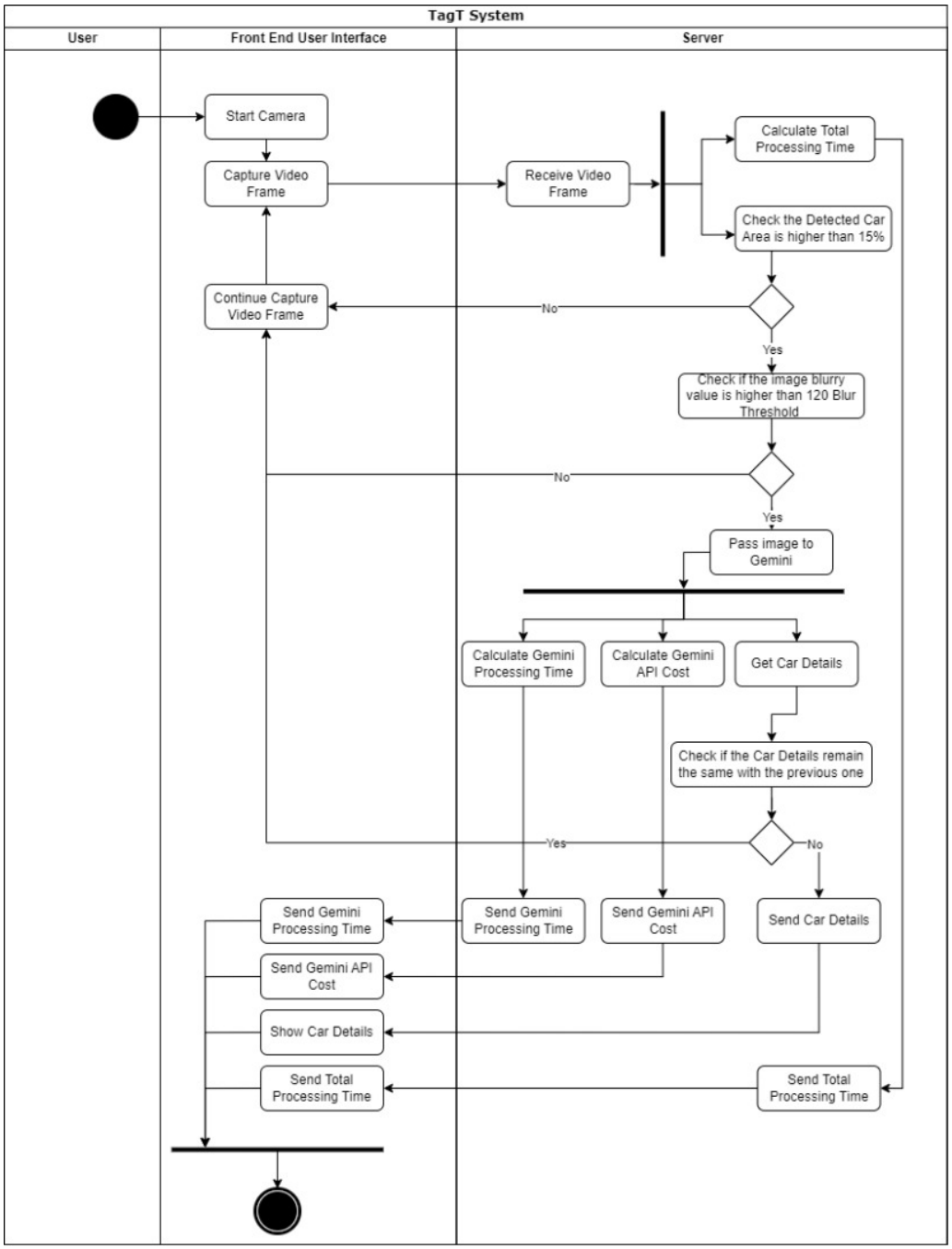


Figure 3.3.1 TagT System Activity Diagram

On the server side, a series of checks are performed to ensure the quality and relevance of the image. First, the system calculates the total processing time and verifies if the detected car area is greater than 15% of the total frame. If this condition is not met, the system continues to capture new video frames. If the condition is met, a subsequent check is performed to determine if the image's blur value is higher than a threshold of 120. Images that are too blurry are discarded, and the system proceeds to capture new frames.

Frames that pass both the area and blur checks are then passed to the Gemini API for detailed analysis. The system calculates the processing time and API cost associated with the Gemini analysis. It then retrieves the car details from the Gemini response. A crucial step follows where the system checks if the newly identified car details are the same as the previous one. If the details are the same, it indicates that the same car is still in the frame, and the system sends the Gemini processing time and API cost to the user interface.

However, if the car details are new, the system sends the car details, along with the Gemini processing time, Gemini API cost, and the total processing time to the front-end user interface. Finally, these details are displayed to the user, and the process concludes. This entire workflow ensures that only high-quality, relevant images are processed, and redundant information is not repeatedly sent, optimizing both performance and cost.

3.4 Use Case Diagram

This section presents the use case diagram for the TagT system, which illustrates the interactions between the user and the system's key functionalities. The diagram models a real-world application, such as at a petrol station, where the system is used to verify vehicle identity to prevent the misuse of subsidized fuel like RON95. Figure 3.4.1 below shows the use case diagram.

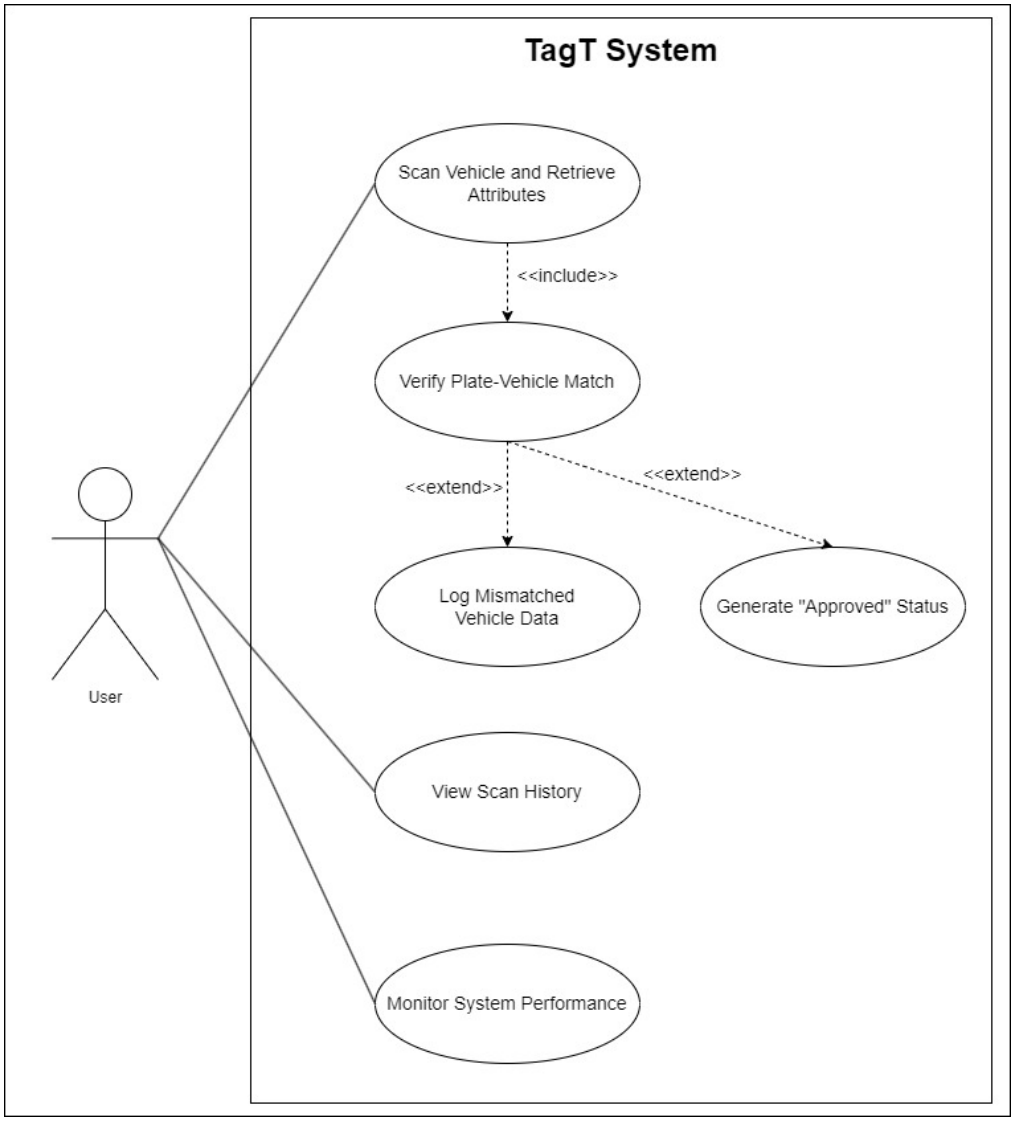


Figure 3.4.1 TagT System Use Case Diagram

Figure 3.4.1 presents the use case diagram, which models the interaction between the User and the TagT system. The user, typically a petrol station attendant, interacts with the system through a mobile application. The system is designed around key use cases that represent the functionalities the user can perform to achieve their goals.

- **Scan Vehicle and Retrieve Attributes**

  This is the foundational use case where the user initiates the process by pointing their mobile device's camera at a vehicle. The system captures the video feed in real-time, sends frames to a back-end server and processes them using YOLO11n-seg for detection and Gemini model for attribute recognition. The system then returns the detected license plate number, car brand and car colour to the user's interface.

- **Verify Plate-Vehicle Match**

  The central goal of the user is to verify if the physical license plate on a vehicle legitimately belongs to it. This use case allows the user to confirm that the car's brand and colour, as identified by the system, are consistent with the license plate number. The outcome of this verification dictates the subsequent actions.

- **Log Mismatched Vehicle Data**

  This use case represents a critical security function. If the verification process reveals a discrepancy. For example, the license plate is registered to a Proton but is attached to a Honda. Then the system provides the user with an option to log the incident. This creates a permanent record of the mismatched data, including the captured image and detected attributes, for future review or action.

- **Generate "Approved" Status**

  This use case represents the successful outcome of the verification process. If the system confirms that the license plate number and the vehicle's attributes are a correct match, it will display a clear visual confirmation, such as an "Approved" status. This signal informs the user that the vehicle is legitimate and they can proceed with the real-world action of allowing the driver to pump RON95 fuel.

- **View Scan History**

  The user can access a comprehensive log of all past scans performed by the system. This history would include details of each vehicle and the timestamp of the scan, which has provided a valuable tool for auditing and record-keeping.

- **Monitor System Performance**

  The system allows the user to view operational metrics for each scan. This includes data such as the total processing time, the Gemini API processing time and the API cost. This functionality is essential for administrative oversight, ensuring the system remains efficient and cost-effective.

The diagram also illustrates the logical flow and dependencies between use cases using UML relationships. The <<include>> relationship between "Verify Plate-Vehicle Match" and "Scan Vehicle and Retrieve Attributes" signifies that a vehicle scan is a mandatory prerequisite for any verification attempt. Furthermore, the <<extend>> relationships show that both "Log Mismatched Vehicle Data" and "Generate 'Approved' Status" are optional outcomes that extend the "Verify Plate-Vehicle Match" use case. These actions are conditional and mutually exclusive; the system will only trigger one based on the result of the verification.

CHAPTER 4 SYSTEM DESIGN

This chapter details the evidence-based design process that led to the final architecture of the TagT system. It begins with a comprehensive investigation into various technologies, presenting the experimental results that justify the selection of each final system component. Following this, the chapter specifies the chosen components and details how they interact.

4.1 Preliminary Investigation and Component Selection

Before finalizing the system architecture, a detailed investigation was conducted to evaluate multiple technologies for each required task. The goal of this preliminary work was to identify the most accurate, efficient and robust components for the system.

Two primary design philosophies were explored:

1. A traditional modular pipeline using single-task models.
2. A modern and unified approach using an advanced AI model.

4.1.1 Evaluation of Single Task Model Approach

4.1.1.1 License Plate Detection

Ten models from Roboflow were tested in both ideal and challenging scenarios such as nighttime, raining days, foggy days and different angle, in order to accurately get the results respectively. The results, shown in Figure 4.1.1.1.1, revealed a range of outcomes across the models, highlighting their strengths and limitations. For example, the “anpr-w2b2/model/384” model achieved 90.00% accuracy in ideal conditions but dropped significantly to 74.00% in various conditions, indicating its sensitivity to environmental factors. Moreover, the “yolov7-license-plate-detection/model/3” model achieved 84.00% accuracy in ideal conditions and 82.00% in various conditions, showing more consistency but still falling short of optimal performance for real-time applications.

In contrast, the “license-plate-detection-merged-projects/model/3” model excelled with 90.00% accuracy in ideal conditions and an impressive 94.00% in various conditions, positioning itself as one of the top performers. Another strong competitor was the “car-plate-detection-sctyn/model/3” model, achieved 92.00% accuracy in ideal conditions and 96.00%

accuracy in various conditions, showcasing its ability across various scenarios. In short, after testing all ten models, the best model for license plate detection was determined to be "car-plate-detection-sctyn/model/3", due to its superior accuracy and reliability across both ideal and various conditions.

	License Plate Detection Model		
	Source	Accuracy	Various Condition Accuracy
1	anpr-vv2b2/model/384	90.00%	74.00%
2	yolov7-license-plate-detection/model/3	84.00%	82.00%
3	car-license-fj1kd/model/4	88.00%	88.00%
4	anpr-code/model/1	84.00%	86.00%
5	license-plate-detection-merged-projects/model/3	90.00%	94.00%
6	license-plate-poc/model/1	86.00%	82.00%
7	car-plate-detection-sctyn/model/3	92.00%	96.00%
8	number-plate-detection-dqbes/model/1	90.24%	92.68%
9	platesv2-0nqdl/model/1	90.00%	76.00%
10	license-plates-jwp4u/model/1	88.00%	82.00%

Figure 4.1.1.1.1 Results from License Plate Detection Models

4.1.1.2 License Plate Number Recognition

Optical Character Recognition (OCR) is used to covert images of text into a machine-readable text format. After selecting the license plate detection model, the next step is to test OCR models for reading the license plate number. There are two OCR models - EasyOCR and TesseractOCR being used to evaluate. Their performance was measured through key metrics such as reading accuracy, precision, recall, and F1-score by using a dataset of license plate images.

The results of EasyOCR revealed significant limitations in its ability to accurately read license plate numbers as shown in Figure 4.1.1.2.1. These results suggest that EasyOCR struggled in reading license plate characters from license plate model detection screenshots, possible due to factors such as varying image quality, font styles, or environmental conditions, leading to a high number of false positives and unreliable text extraction. On the other hand, TesseractOCR performed even worse in the license plate reading task.

In conclusion, the performance of EasyOCR and TesseractOCR in reading license plate numbers showed significant shortcomings as neither achieving the level of accuracy or

reliability for practical deployment. These results highlight the need for more advanced OCR techniques or enhanced preprocessing methods to improved text reading accuracy.

```
image_path,true_plate_text,predicted_plate_text
TP10.JPEG,PRF5204,5204PRF
TP11.JPEG,PDW128,PW128
TP12.JPEG,VHX2887,WH2882
TP13.JPEG,PGH171,POHIZ
TP14.JPEG,PPN5652,PPN5652
TP15.JPEG,PLN577,PLN5Z7
TP16.JPEG,JGF1331,JGHIB31
TP17.JPEG,VHQ1269,1289VH@
TP2.JPEG,PMS8389,RHS8389
TP3.JPEG,QAB203C,QAB2036
TP4.JPEG,PQX8362,POX8362
TP5.JPEG,KFQ6185,Unknown
TP6.JPEG,VCV2736,VCV2736
TP7.JPEG,PLQ8589,RU8589
TP8.JPEG,PGS811,PESG
TP9.JPEG,PJY6713,PY6Z8
```

Figure 4.1.1.2.1 Inaccurate reading from easyOCR

4.1.1.3 Car Brand Detection and Recognition

Following the evaluation of the license plate detection and license plate number reading models, the next step shifted to testing Roboflow models for car brand detection and recognition, which solely focus on a specific set of car brands which are Honda, Mazda, Perodua, Proton and Toyota. A total of six models were evaluated and their performance was measured based on accuracy and the range of brands they could recognize. The results, as shown in Figure 4.1.1.3.1, showed both their potential and their limitations in addressing the project's requirements for reliable car brand recognition.

The evaluation results highlighted a wide range of performance among the tested models. The “car-models-ves3u/1” model achieved the lowest accuracy at 54.00%, while the “carbrand5001/model/1” model achieved a moderate 70.00% accuracy. The “walao/4” and “walao/5” models performed better, with accuracies of 82.00% and 86.67% respectively. Those models were all capable of recognizing the selected brands. Meanwhile, the “car-logo-cyxpe/model/10” and “car-logo-detection-2-2xc2d/1” models were the top performers as both achieving a perfect 100.00% accuracy. However, “car-logo-cyxpe/model/10” could only recognize four of the five targeted brands, excluding Mazda, and “car-logo-detection-2-2xc2d/1” faced technical issues such as the “Request Entity Too Large” error for URLs. This exposed a significant limitation in their flexible for real-world applications.

In short, the testing of Roboflow models for car brand detection and recognition provided valuable insights into their strengths and constraints. Although there were models that achieved 100% accuracy in car brand recognition, they were not capable in real-world situations as they cannot recognize the other car brands such as Audi, BMW, Mitsubishi and the others. Thus, in order to achieve the project's goal of reliable and inclusive car brand detection in dynamic environments, expanding the brand recognition capabilities of the top-performing models or by improving the accuracy of models with broader coverage is needed.

	Car Brand Recognition Model		
	Source	Accuracy	Recognition Car Brand
1	car-models-ves3u/1	54.00%	Honda, Mazda, Perodua, Proton, Toyota
2	walao/4	82.00%	Honda, Mazda, Perodua, Proton, Toyota
3	car-logo-cyxpe/model/10	100.00%	Honda, Perodua, Proton, Toyota
4	car-logo-detection-2-2xc2d/1	100.00%	Honda, Mazda, Perodua, Proton, Toyota
5	walao/5	86.67%	Honda, Mazda, Perodua, Proton, Toyota
6	carbrand5001/model/1	70.00%	Honda, Mazda, Perodua, Proton, Toyota

Figure 4.1.1.3.1 Results from Car Brand Recognition Models

4.1.1.4 Car Detection

Before initiating testing for the car colour recognition model, we focused on testing YOLO models for car detection first. Five YOLO models, ranging from YOLOv8n to YOLO12n were evaluated and their performance was measured based on accuracy and processing time in both ideal and challenging conditions, such as nighttime, raining days, foggy days and side view of the car. The results, as presented in Figure 4.1.1.4.1, provided a comprehensive view of each model's capabilities, emphasizing their strengths and trade-offs in meeting the project's requirements for real-time car detection.

The results show that all YOLO models performed well in terms of accuracy, with varying degrees of efficiency in processing time. The YOLOv8n model achieves a 94.00% accuracy in ideal conditions, taking 5.22 seconds, but its accuracy drops to 84.00% in various conditions, with a slightly reduced processing time of 4.76 seconds. Similarly, YOLOv9t and YOLOv10n both recorded a higher accuracy of 98.00% in ideal conditions and 84.00% in challenging conditions. Although they have same accuracy in ideal and challenging conditions, YOLOv10n is faster than YOLOv9t. In addition, the YOLO11n model shows a balanced performance as it achieves 98.00% accuracy in ideal conditions within 4.45 seconds and 88.00% in various

conditions within 4.00 seconds, making it one of the faster models. The top performer in terms of accuracy is YOLO12n, achieves a perfect 100.00% accuracy in ideal conditions within 5.94 seconds, but its accuracy in various conditions is only 84.00% with a 4.81 second processing time.

Overall, the testing of YOLO models for car detection demonstrated their strong potential for accurate car detection and YOLO11n was selected as the most suitable model on car detection.

	Car Detection YOLO Model				
	Model	Accuracy	Time Used	Various Condition Accuracy	Time Used
1	YOLOv8n	94.00%	5.22 seconds	84.00%	4.76 seconds
2	YOLOv9t	98.00%	6.35 seconds	84.00%	6.17 seconds
3	YOLOv10n	98.00%	5.52 seconds	84.00%	3.84 seconds
4	YOLO11n	**98.00%**	**4.45 seconds**	**88.00%**	**4.00 seconds**
5	YOLO12n	100.00%	5.94 seconds	84.00%	4.81 seconds

Figure 4.1.1.4.1 Results from Yolo Models on Car Detection

4.1.1.5 Car Colour Recognition

Once the best model for car detection was identified, we can integrate YOLO11n to detect and capture images of cars which were then used to test models for car colour recognition. There were three different methods to recognize car colour which are using the ResNet18 model, an HSV-based method, and K-means clustering. The performance of each method was assessed based on accuracy and the results presented in Figure 4.1.1.5.1.

The evaluation results highlighted a significant difference in performance among the tested methods. The ResNet18 model achieved the highest accuracy at 81.82%, demonstrating its ability to distinguish car colours compared to the other approaches. In contrast, the HSV-based method achieved accuracy of 65.91% and K-means clustering achieved accuracy of 56.82%, indicating that they struggled with consistency, likely due to its reliance on colour space transformations or its limitations in accurately grouping colours in a way that aligns with human perception.

To conclude, the testing of methods for car colour recognition emphasized the potential of the ResNet18 model as the most reliable approach. However, to meet the project's objective, ResNet18 model needs to be optimized for better performance.

	Car Colour Recognition Model	
	Method	Accuracy
1	ResNet18 model	81.82%
2	HSV-based	65.91%
3	K-means clustering	56.82%

Figure 4.1.1.5.1 Results from Car Colour Recognition Methods

The results from this modular approach revealed a critical insight; while some individual components performed well, the pipeline as a whole was fragile. Key weaknesses in OCR and the inflexibility of the brand recognition models made this approach unsuitable for achieving the project's goals.

4.1.2 Evaluation of Advanced AI Models

As we can see some attribute recognition models did not perform well and struggled to achieve high accuracy, advanced AI models such as Gemini, Grok, ChatGPT, and Qwen are used to test car attribute recognition. The evaluation focused on their ability to recognize license plate numbers, car brands and colours, with performance assessed based on frames resolution, total frames per second, processing time, and cost usage. The results, show from Figure 4.1.2.1 to Figure 4.1.2.6, provide valuable insights into their effectiveness for real-time car attribute recognition.

First, ChatGPT and Qwen were removed from evaluation due to their consistently unreliable outputs, even after adjusting the input prompts. The correct car details being tested was “Honda” as car brand; “White” as car colour; “PNN1678” as license plate number. However, ChatGPT was tested at 1280x720 pixels with 20 frames and identified “Honda” and “White” correctly but gave inconsistent plate readings like “WGB 6188” and “PWN 1678”. Qwen was getting worse as it tested at 640x480 pixels with 25 frames and predicted varying brands which are “Honda”, “Nissan”, and “Ford”, colours which are “Red”, “Black”, and “Blue”, and plates which are “V345ABC”, “T678DEF”, and “K456GHI”, making both unsuitable for the project.

With ChatGPT and Qwen excluded, the testing focused on Gemini and Grok. At first, Gemini and Grok were tested with a video in the first one second at a resolution of 640x480 pixels and 20 frames per second (FPS). Gemini identified the car brand as “Honda”, colour as “White”,

and the plate as “PAN1678”, with a processing time of 3.78 seconds and a cost of $0.000329; while Grok, under the same conditions, also recognized “Honda” and “White” and the plate reading as “PNP1678”, taking 9.33 seconds and at a cost of $0.001065. Due to both models could not recognize the plate number correctly, increasing the frame rate to 25 FPS at the same resolution is needed. Then, both models have correctly identified the plate as “PNN1678” alongside “Honda” and “White”, with Gemini at 3.45 seconds and $0.000410, and Grok at 9.50 seconds and $0.001328.

Next, increase the resolution to 1280x720 pixels while maintaining 25 FPS, Gemini retained accurate results at 6.51 seconds and $0.000873, while Grok could not achieve accurate result. From the Figure 4.1.2.2, show that Grok could not achieve accurate result when the frames resolution is higher than 640x480 pixels. Nevertheless, Gemini still retained accurate results even though the resolution is 1280x720 pixels with 5 FPS, by using 2.45 seconds and $0.000199. Moreover, Gemini can achieve accurate result when the resolution is 320x240 pixels and 5 FPS, by using 1.76 seconds and $0.000049, as show in Figure 4.1.2.1. This effectively reduces the time used and cost effective.

	Gemini without YOLO									
Resolution	640x480	640x480	1280x720	1280x720	1920x1080	1920x1080	320x240	320x240	240x180	160x120
Total frames per second	20	25	25	5	20	25	25	5	30	30
Cost	$0.000329	$0.000410	$0.000873	$0.000199	$0.001180	$0.001488	$0.000158	$0.000049	$0.000133	$0.000082
Gemini Return Time	3.78s	3.45s	6.51s	2.45s	4.83s	8.97s	2.60s	1.76s	2.61s	2.48s
Result - Brand	Honda	Honda	Honda	Honda	Honda	Honda	Honda	Honda	Honda	Honda
Result - Colour	White	White	White	White	White	White	White	White	White	White
Result - Plate Number	PAN1678	PNN1678	PNN1678	PNN1678	PAN1678	PNN1678	PNN1678	PNN1678	PMN1678	PBN578
Status	Incorrect	**Correct**	**Correct**	**Correct**	Incorrect	**Correct**	**Correct**	**Correct**	Incorrect	Incorrect

Figure 4.1.2.1 Results from Gemini

	Grok without YOLO								
Resolution	640x480	640x480	1280x720	1920x1080	320x240	320x240	240x180	240x180	160x120
Total frames per second	25	20	25	25	25	20	25	20	30
Cost	$0.001328	$0.001065	$0.002856	$0.004880	$0.000515	$0.000422	$0.000357	$0.000297	$0.000251
Grok Return Time	9.50s	9.33s	13.56s	14.50s	6.57s	5.24s	5.55s	4.16s	3.26s
Result -Brand	Honda	Honda	Honda	Honda	Honda	Honda	Honda	Honda	Honda
Result -Colour	White	White	White	White	White	White	White	White	White
Result -Plate Number	PNN1678	PNP1678	PNN1519	PNN1519	PNN1678	PAB1678	PNN1678	PNB1678	PWD 6718
Status	**Correct**	Incorrect	Incorrect	Incorrect	**Correct**	Incorrect	**Correct**	Incorrect	Incorrect

Figure 4.1.2.2 Results from Grok

As using Gemini and Grok require API cost, the way to reduce numbers of frames before sending to Gemini and Grok is needed. Hence, YOLO was initially integrated to address this issue. According to Figure 4.1.2.3 and Figure 4.1.2.4, 16 frames at 640x480 pixels resolution were sent to Gemini and Grok and both of them have accurately identified “Honda”, “White”, and “PNN1678”. Gemini took 3.68 seconds and used $0.000268, while Grok took 6.83 seconds and used $0.000879. The results show a reduction in frames processed and costs compared to

the results without using YOLO models. Alternatively, Grok could not correctly recognize the plate number once the resolution has decreased. However, Gemini still achieved correct recognition when the resolution has been decreased to 320x240 pixels and only 5 frames were being sent to Gemini, by using 2.28 seconds and a cost of $0.000052.

	Gemini with YOLO			
Resolution	640x480	1280x720	1920x1080	320x240
Frame Per Second	25	5	25	5
Numbers of Frame with Car Detected	16	5	16	5
Cost	$0.000268	$0.000195	$0.000963	$0.000052
Gemini Return Time	3.68s	2.96s	8.24s	2.28s
Result - Brand	Honda	Honda	Honda	Honda
Result - Colour	White	White	White	White
Result - Plate Number	PNN1678	PNN1678	PNN1678	PNN1678
Status	**Correct**	**Correct**	**Correct**	**Correct**

Figure 4.1.2.3 Results from Gemini and YOLO

	Grok with YOLO			
Resolution	640x480	320x240	320x240	240x180
Frame per second	25	25	30	25
Numbers of Frame with Car Detected	16	16	18	16
Cost	$0.000879	$0.000355	$0.000391	$0.000253
Grok Return Time	6.83s	5.24s	5.57s	2.66s
Result -Brand	Honda	Honda	Honda	Honda
Result -Colour	White	White	White	White
Result -Plate Number	PNN1678	PNP1678	PNP1678	PNP1678
Status	**Correct**	Incorrect	Incorrect	Incorrect

Figure 4.1.2.4 Results from Grok and YOLO

To further reduce processing time and costs, ResNet18 was integrated and aimed to minimize number of frames before performing attribute recognition. ResNet18 used cosine similarity with a threshold of 0.95 to evaluate the results. Based on Figure 4.1.2.5, Gemini has achieved good result when the resolution was at 320x240 pixels and 4 frames were being sent to Gemini, by using 1.92 seconds and at a cost of $0.000049. While Grok could not accurately recognize the plate number even though the cosine similarity threshold has changed to 0.97, as shown in Figure 4.1.2.6.

In short, Gemini showed better consistency and efficiency with YOLO and ResNet18, offering the best balance of accuracy, speed, cost, and robustness across different conditions.

	Gemini with YOLO and ResNet18						
Resolution	640x480	1280x720	1280x720	1920x1080	320x240	320x240	320x240
Confidence Threshold	0.95	0.95	0.95	0.95	0.95	0.95	0.95
Frame Per Second	25	5	10	25	5	10	15
Numbers of Frames Left before Recognition	4	3	4	4	3	4	4
Cost	$0.000086	$0.000127	$0.000162	$0.000257	$0.000043	$0.000051	$0.000049
Gemini Return Time	5.13s	2.90s	3.55s	4.47s	1.87s	2.05s	1.92s
Result - Brand	Honda	Honda	Honda	Honda	Honda	Honda	Honda
Result - Colour	White	White	White	White	White	White	White
Result - Plate Number	PNN1678	PAN1678	PNN1678	PNN1678	PMN 1678	PMN 1678	PNN 1678
Status	**Correct**	Incorrect	**Correct**	**Correct**	Incorrect	Incorrect	**Correct**

Figure 4.1.2.5 Results from Gemini, YOLO and Resnet18

	Grok with YOLO and ResNet18	
Resolution	640x480	640x480
Confidence Threshold	0.95	0.97
Frame Per Second	25	25
Numbers of Frames Left before Recognition	4	7
Cost	$0.000267	$0.000419
Grok Return Time	5.38s	2.99s
Result -Brand	Honda	Honda
Result -Colour	White	White
Result -Plate Number	PNP1678	PNP1678
Status	Incorrect	Incorrect

Figure 4.1.2.6 Results from Grok, YOLO and Resnet18

4.1.3 Final Design Decision

The extensive preliminary work demonstrated that a traditional, multi-model pipeline was not viable due to critical weaknesses in areas like OCR and the inflexibility of specialized car brand recognition models. Instead, a hybrid architecture was determined to be the optimal solution. This final design leverages the YOLO11n model for fast and efficient car detection and the Gemini model for accurate and robust multi-attribute recognition.

Furthermore, the preliminary investigation revealed a critical trade-off between input resolution, cost and recognition reliability. While a lower resolution of 320x240 offered the fastest processing times and lowest costs in Gemini, its accuracy was inconsistent across tests. However, a resolution of 640x480 was found to provide a more robust and reliable input for the Gemini model, consistently yielding correct results.

Therefore, the final system design incorporates not only the selected models but also a key configuration directive: all images before sending to the Gemini API are to be standardized to a 640x480 resolution. This approach combines the strengths of both technologies and configures them in a way that prioritizes accuracy and reliability, creating a system that is both effective and efficient.

4.2 System Block Diagram

Based on the findings from the preliminary investigation, a hybrid client-server architecture was designed. This section presents the system block diagram, which provides a high-level overview of the final system's structure and the flow of data between its major components. The system is designed to capture and process real-time video frames from a mobile device to accurately identify a car's license plate number, car brand and colour. It consists of three primary components: a user, a mobile device application and a back-end server. Each component within the components are performing a specific role within the data capture and analysis pipeline. Figure 4.2.1 below illustrates the system block diagram of the TagT system.

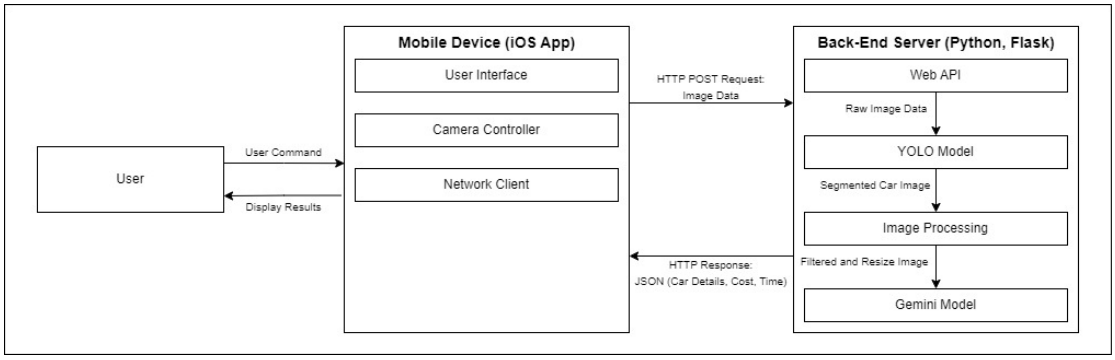


Figure 4.2.1 TagT System Block Diagram

Based on Figure 4.2.1, the process begins with the User, who initiates a scan by sending a "User Command" to the Mobile Device (iOS App). Within the app, the Camera Controller captures a high-resolution image from the video feed. This image is then handled by the Network Client, which prepares and transmits the image data to the Back-End Server via an HTTP POST Request.

The Back-End Server, built with Python and Flask, receives the request at its Web API endpoint. The "Raw Image Data" is passed directly to the YOLO Model. YOLO model is responsible to perform car detection and segment out the car from the full image, producing a "Segmented Car Image" that isolates the car from its background.

This segmented image is then forwarded to the Image Processing module. At this stage, quality checks such as filtering for blurriness and ensuring the car's area is sufficient are performed on the smaller, cropped image for maximum efficiency. The image is also resized to a standard

resolution, resulting in a "Filtered and Resize Image". This final, optimized image will be send to the Gemini Model. The advanced vision model is responsible to analyse the image to recognize and extract the required attributes: license plate number, car brand and colour.

Once the analysis is complete, the back-end server compiles the extracted attributes, along with performance metrics like API cost and processing time into a structured format. This data is sent back to the Mobile Device's Network Client as an HTTP Response containing a JSON payload. Finally, the User Interface on the mobile app parses this data and updates the screen to "Display Results" to the user in real-time.

This client-server architecture enables the system to leverage a lightweight mobile front-end for user interaction and data capture, while offloading all computationally intensive AI processing to a more powerful back-end. This design provides a scalable and efficient solution for real-time car identification.

4.2.1 Breakdown of System Block Diagram

To better understand how each component functions within the system, the key modules from the system block diagram are summarized below:

- **User**

  The User is the primary actor who interacts with the system. They initiate the car identification process by issuing a command through the mobile application and view the final results on the device's screen.

- **Mobile Device (iOS App)**

  The mobile device serves as the front-end user interface for the TagT system. It is responsible to let user interacts with it and capture the real-time image data. It contains three core software components:

  - **User Interface**

    The User Interface is built with Apple's SwiftUI framework, the UI provides the real-time camera preview, control buttons for starting and stopping the analysis, and a clear display for the final results, including the vehicle's attributes and performance metrics.

  - **Camera Controller**

    The Camera Controller uses the native AVFoundation framework to manage the device's camera. It is responsible for configuring the video capture session and providing the high-resolution image frame that is used for analysis.

  - **Network Client**

    The Network Client handles all communication with the back-end server. It takes the image frame captured by the camera, packages it into an HTTP POST request and transmits it to the server. It also receives the JSON response from the server and passes the data to the User Interface for display.

- **Web API**

  The Web API is built with Python and the Flask framework, serves as the entry point for the back-end server. It listens for incoming HTTP requests from the mobile client, validates that an image file is present and passes the raw image data into the processing pipeline. At the end of the pipeline, it formats the final results into a JSON object and sends it back to the client as the HTTP response.

- **YOLO Model**

  Upon receiving the raw image data from the Web API, the YOLO11n-seg model is the first processing stage. Its primary function is to perform efficient object detection on the entire image to locate any cars. Once a car is identified, the model generates a segmentation mask to precisely crop the car from its background, ensuring that subsequent processing stages only focus on the relevant object.

- **Image Processing Module**

  This module receives the segmented car image from the YOLO model and is responsible for quality control and standardization. It performs two key filtering checks: it calculates the image's Laplacian variance to discard blurry images and verifies that the car's area is above a minimum percentage threshold. If the image passes these checks, it is resized to a standard resolution of 640x480 pixels using OpenCV. This ensures that the input for the final recognition model is consistent, which improves both performance and accuracy.

- **Gemini Model**

  The filtered and resized image is finally passed to the Gemini 1.5 Flash model for attribute recognition. This multimodal generative AI model analyses the visual content of the image based on a specific text prompt. Its task is to identify and extract the car's key attributes: license plate number, car brand and colour. The structured data extracted by this model constitutes the final output of the analysis pipeline.

4.3 System Components Specifications

This section provides a detailed breakdown of the hardware and software components used in the final implementation of the TagT system.

4.3.1 Hardware Components

To implement the system, two main hardware components are required:

- **Mobile Device**
  An iPhone is used as the front-end client. It runs the mobile application, uses its camera to capture the live video feed and displays the final analysis results and system performance metrics to the user.

- **Back-end Server**
  A personal computer is used to host the back-end system. The machine runs the Python API and performs all the heavy processing tasks, including car detection, image quality checks and car attribute recognition using the deployed AI models. It receives images from the mobile app and sends back a JSON response with the results.

4.3.2 Software Components

The software architecture consists of a front-end application and a back-end server that communicate over a network. The key software components are as follows:

- **SwiftUI (Front-end User Interface)**
  The mobile application is built as a native iOS app using Apple's SwiftUI framework. It provides the complete user interface, which includes the live camera view, start and stop controls, and the display that shows the detected car details, scan history and processing statistics.

- **AVFoundation (Camera Control)**
  Within the iOS application, the AVFoundation framework is used to access and control the device's camera. It manages the live video capture session and provides the image frames that are sent to the back-end server for analysis.

- **Python with Flask (Back-end API)**

  The back-end is a web API created using Python and the Flask framework. It provides a simple endpoint that receives image files from the front-end app via HTTP requests. Flask handles the network communication, routes the data to the processing script and returns the final results.

- **YOLO11n-seg (Car Detection)**

  The YOLO11n-seg model is used on the server to perform initial car detection and segmentation. When the back-end receives an image, this model identifies if a car is present and creates a segmentation mask to isolate the car from its background, ensuring only the relevant part of the image is analyzed further.

- **Google Gemini 1.5 Flash (Car Recognition)**

  The segmented image of the car is then sent to the Gemini 1.5 Flash model. This generative vision model analyzes the image and based on a specific prompt, it extracts the car's attributes: license plate number, car brand and colour.

- **OpenCV and NumPy (Image Processing)**

  These Python libraries are used for various image manipulation tasks on the back-end. OpenCV is used to decode the image, rotate it correctly and calculate its blurriness through Laplacian variance for quality filtering. NumPy is used for efficient numerical operations on the image data arrays.

CHAPTER 5 SYSTEM IMPLEMENTATION

5.1 Hardware Setup

The implementation and operation of the TagT system rely on two primary hardware components: a back-end server for development and processing, and a mobile client device for real-time operation. This section details the specifications of the hardware used for this project.

5.1.1 Back-End Development Server

The back-end server is the machine where the Python-based API was developed, tested and run. It is responsible for handling all computationally intensive tasks, including running the YOLO and Gemini models for car analysis. The specifications for the development server are provided in Table 5.1.1.1.

Description	**Specifications**
Model	Lenovo IdeaPad 5 Pro 16ARH7
Processor	AMD Ryzen 7 6800HS
Operating System	Windows 11
Graphic	NVDIA GeForce RTX
Memory	16GB RAM
Storage	474GB

Table 5.1.1.1 Specifications of the Back-End Development Server

5.1.2 Front-End Client Device

The front-end client is the mobile device used for the real-world deployment and operation of the TagT system. It is responsible for running the native iOS application, capturing the live video feed via its camera and displaying the final analysis results to the user. The specifications for the mobile device are provided in Table 5.1.2.1.

Components	**Specifications**
Model	iPhone 12 Pro
Operating System	iOS 18.6.2
System Chip	A14 Bionic Chip
Storage	128GB
RAM	6GB

Table 5.1.2.1 Specifications of the Front-End Client Device

5.2 Software Setup

This section describes the software environments, libraries and frameworks used to implement the TagT system. The implementation is divided into two main components: a native iOS front-end for user interaction and video capture and a Python-based back-end for car detection and recognition.

5.2.1 Back-End Environment Setup

The back-end was implemented as a real-time processing service written in Python. It runs on a local server and is responsible for receiving an image from the mobile device, performing all AI-driven analysis and returning the structured results.

Key software components used in the back-end environment include:

- Flask for creating the web API connect to the front-end.
- Ultralytics (YOLO) for load and run the yolo11n-seg.pt model for car detection and segmentation.
- google-generativeai (Gemini) for interacting with the Gemini 1.5 Flash API for car attribute recognition.

- OpenCV for image filtering operations.
- NumPy for image resizing operations.

The back-end workflow begins when the Flask API receives an image. This image is immediately passed to the YOLO model to detect and segment a car. The resulting cropped image is then processed by OpenCV to check its quality and resize it to a standard resolution. Finally, the prepared image is sent via the Google client library to the Gemini model for analysis. The extracted details are formatted into a JSON object by Flask and returned to the front-end. This server-side architecture offloads all heavy computation, allowing the mobile client to remain lightweight and responsive.

5.2.2 Front-End Environment Setup

The front-end is a native iOS application developed in Swift using Apple's Xcode IDE. The application provides the user interface, manages the device's camera, and handles all communication with the back-end server.

Core software frameworks used in the front-end environment include:

- SwiftUI for building the user interface, including the live camera view, control buttons and results display.
- AVFoundation for accessing and managing the device camera for capturing high-resolution video frames.
- Foundation for handling all network communication.

When the user initiates a scan via the SwiftUI interface, the AVFoundation framework captures the current video frame. This image is then sent to the back-end server using a URLSession HTTP request. The application waits for the server to respond with a JSON payload, which it then decodes into a Swift data structure. This data is used to update the SwiftUI views in real-time, displaying the identified license plate number, car brand and colour to the user. This native application design ensures a smooth user experience and efficient use of the device's hardware.

5.3 Setting and Configuration

This section describes the specific settings and configuration parameters required for the successful operation of the TagT system. These settings control the behaviour of the image processing pipeline, the AI models and the communication between the front-end client and the back-end server.

5.3.1 Back-End Server Configuration

The Python-based back-end server is configured through a set of constants defined directly in the main application script. Upon execution, the server logs its complete initialization sequence to the console. This log, shown in Figure 5.3.1.1, confirms that all key configurations are loaded and AI models are ready before the server begins listening for requests.

```
==================================================
 T A G T  S E R V E R  I N I T I A L I Z A T I O N
==================================================
⚙️ Loading server configuration...
    - MIN_CAR_AREA_PERCENT: 15%
    - BLUR_THRESHOLD:       120.0
    - TARGET_RESOLUTION:    640x480
    - GEMINI_PROMPT:        'Extract motor vehicles only. Format: * Brand: [bra...'
✅  Configuration loaded.
--------------------------------------------------
⚙️ Loading YOLO11n-seg model from local file...
✅  YOLO model loaded successfully.
⚙️ Configuring Gemini 1.5 Flash model via API...
✅  Gemini model configured successfully.
--------------------------------------------------
🚀 All components initialized. Launching Flask server...
    - Listening on: http://0.0.0.0:5000
    - NOTE: Use your ngrok URL for the iOS app to connect.
==================================================
```

Figure 5.3.1.1 TagT Server Initialization

As illustrated in the log, the startup process displays the critical operational parameters that have been configured. It then confirms the successful loading of the local YOLO model and the initialization of the Gemini API.

Key configured parameters are as follows:

- MIN_CAR_AREA_PERCENT = 15

  A vehicle must occupy at least 15% of the total image area.
- BLUR_THRESHOLD = 120.0

  The Laplacian variance of the cropped image must be above this value.

- TARGET_CROP_RESOLUTION = (640, 480)
  Sets the standard image size to 640x480 pixels.
- Gemini Model Prompt
  A specific, structured text prompt is configured to instruct the Gemini model on how to format its response, ensuring the output can be reliably parsed.

5.3.2 Front-End and Network Configuration

The front-end iOS application requires configuration to connect to the back-end server and to access the device's hardware.

- Network Connectivity Configuration
  A key requirement is establishing a stable network connection between the front-end application and the back-end server. The main user interface, shown in Figure 5.3.2.1, provides a real-time status indicator to confirm this connection.

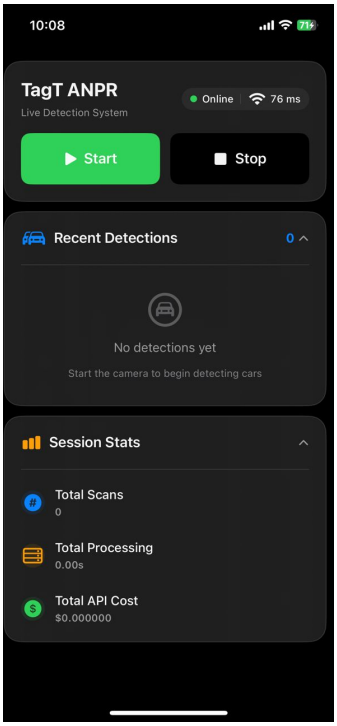


Figure 5.3.2.1 Main Page User Interface with Details

The "Online" status, along with the measured network latency, visually confirms that the front-end has successfully connected to the server's configured baseURL. During development, this was achieved by utilizing ngrok, a reverse proxy service that creates a secure, publicly accessible URL for the local server, which was then configured within the application's networking layer.

- Hardware Access Configuration

  The second essential configuration involves hardware access, which is managed by the iOS operating system. When the "Start" button is pressed for the first time, the application must request permission from the user to access the device's camera, as depicted in the system dialog shown in Figure 5.3.2.2.

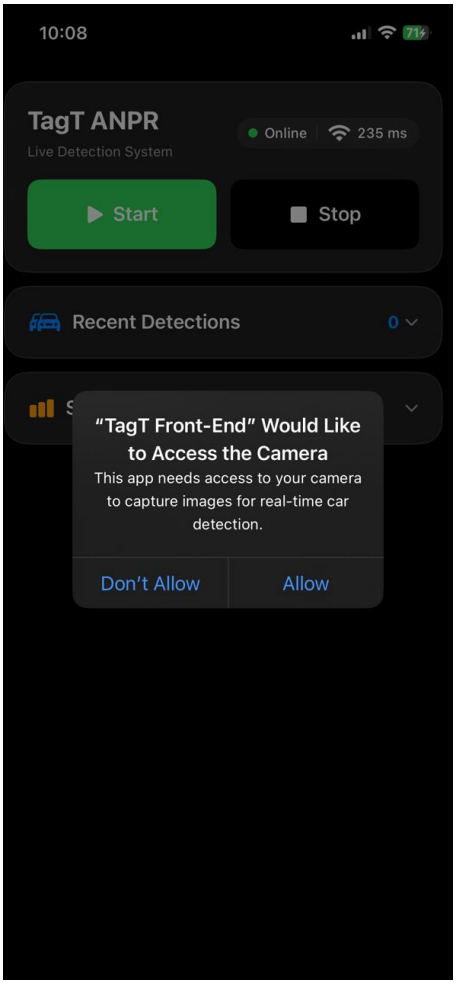


Figure 5.3.2.2 Camera Access

This is a mandatory setup step required by iOS. The user must grant this permission for capturing video frames for analysis. Once permission is granted, it is saved by the operating system for all future sessions.

5.4 System Operation

This section demonstrates the operational workflow of the TagT system from the user's perspective. The process is illustrated with screenshots of the native iOS application, showcasing the user interface at each key stage of a typical analysis cycle, from starting the camera to viewing the final results.

Step 1: Application Launch and System Standby

The system's operation begins when the user launches the application. The main user interface is displayed, as shown in Figure 5.4.1. In this standby state, the application confirms its connection status to the back-end server as it shows "Online" on the top right hand side. The "Recent Detections" and "Session Stats" panels are initially empty.

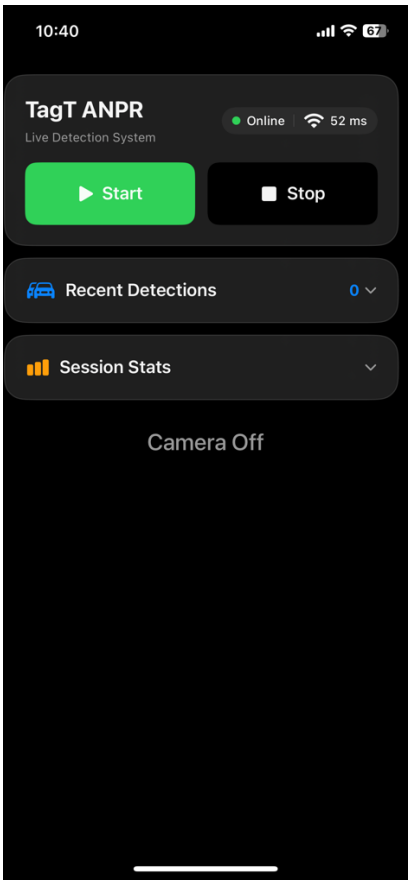


Figure 5.4.1 Main Page User Interface

Step 2: Activating the Live Camera Feed

The user presses the "Start" button to begin the live detection session. The interface transitions to a full-screen, real-time video feed from the device's camera as shown in Figure 5.4.2. In this mode, the application begins to automatically and continuously stream frames to the back-end server for analysis in the background. The user's only task is to keep the target car in the frame.

Figure 5.4.2 Camera Open Interface

Step 3: Autonomous Frame Filtering and Analysis

The back-end server autonomously evaluates each incoming frame against the pre-configured quality thresholds for blurriness and car area. Most frames are instantly discarded. When a high-quality frame that meets the criteria is identified, the system automatically initiates the full analysis pipeline. During this brief processing period, the front-end provides feedback to the user by displaying an "Analysing Car..." indicator as shown in Figure 5.4.3.

Figure 5.4.3 Car is Analysing

Step 4: Displaying a Successful Detection

Once the back-end completes its analysis, which includes segmenting the car as shown in Figure 5.4.4, the results are sent back to the application. The system then updates the UI in two ways:

1. A temporary Results Overlay appears with the identified attributes and performance metrics as shown in Figure 5.4.5.
2. The main user interface is updated in the background with a new entry in "Recent Detections" and revised "Session Stats" as shown in Figure 5.4.6.

The system is designed to intelligently handle continuous video. If the same car remains in view, new overlays will not appear in order to prevent redundant notifications. A new result is only shown when a different car is detected.

Figure 5.4.4 Segmented Car

Figure 5.4.5 Results Display

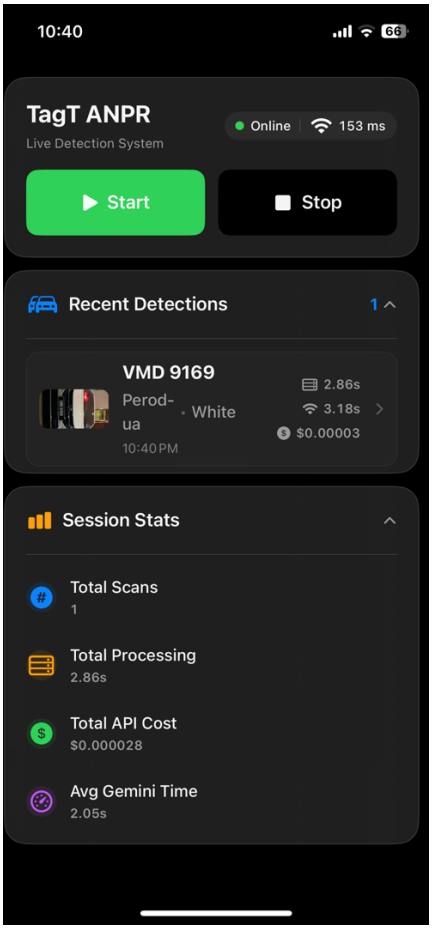


Figure 5.4.6 Car Details stored in Recent Detections

Step 5: Reviewing Scan Details

At any time, the user can press "Stop" to return to the main menu. From there, they can review the session's findings by tapping on an entry in the "Recent Detections" list. This opens a dedicated detail view as shown in Figure 5.4.7, which presents all recorded information for that scan, including a larger thumbnail and a comprehensive list of performance metrics.

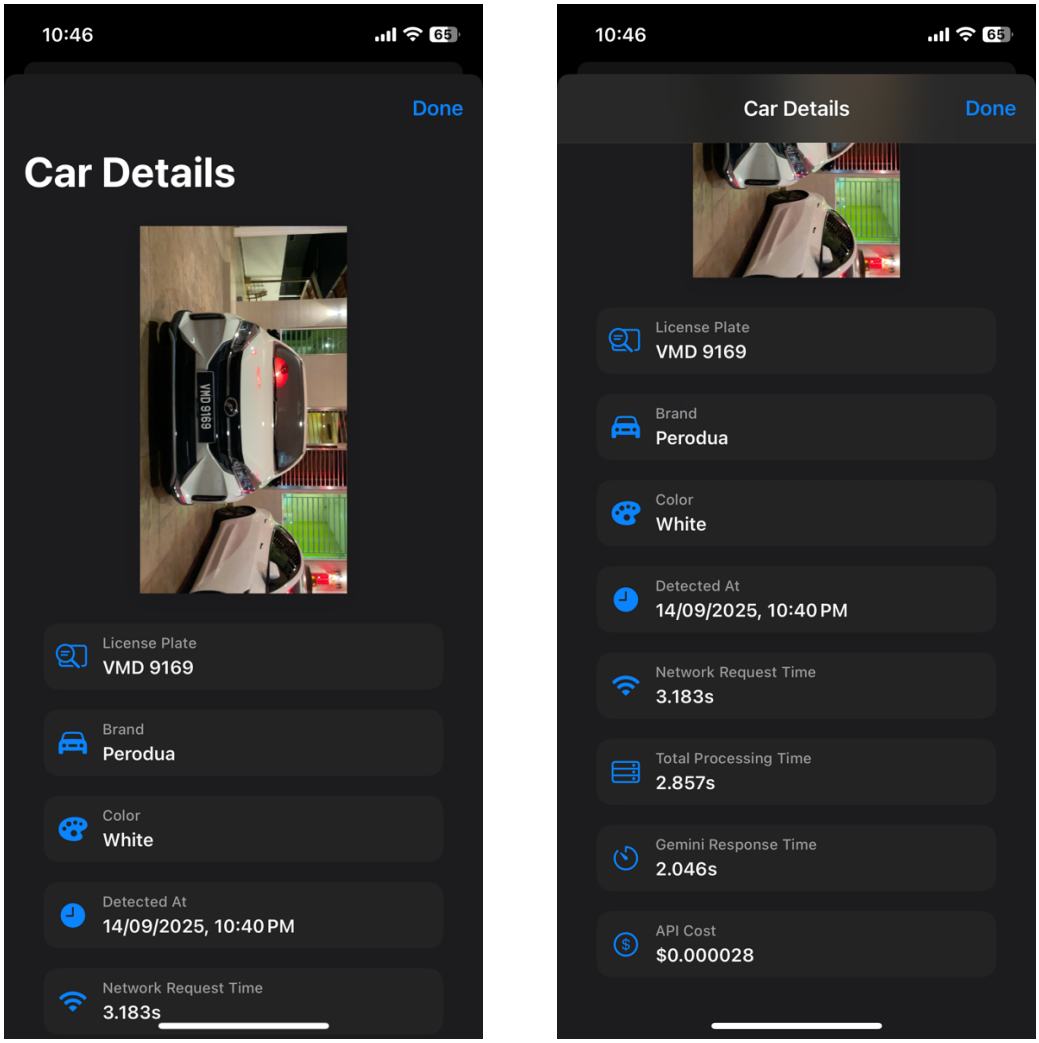


Figure 5.4.7 Car Details User Interface

This automatic operational flow provides a seamless "point-and-shoot" user experience, delivering powerful real-time car analysis without requiring any manual triggering from the user.

5.5 Implement Issues and Challenges

During the development and integration of the TagT system, several technical challenges were encountered. Each issue required a strategic solution, from re-architecting the core logic to implementing specific tools, to ensure the final system was fast, accurate and reliable. The key challenges are detailed below.

- **Failure of Traditional OCR for License Plate Recognition**

  The initial system design depended on a modular pipeline using Optical Character Recognition (OCR) to read license plates. However, both EasyOCR and TesseractOCR models failed to deliver reliable or accurate results from the detected plate images. This poor performance was a critical failure point, making the entire modular approach unviable. This challenge was overcome by fundamentally re-architecting the system to use the Gemini model, whose powerful multimodal capabilities could perform recognition without a separate, fragile OCR step.

- **Inflexible and Unscalable Brand Recognition**

  The second weakness in the modular approach was the car brand recognition models. While some pre-trained models from Roboflow were accurate for a small, specific set of brands, they were unable to recognize any car outside their limited training data. This lack of scalability made them unsuitable for a real-world application. The pivot to the Gemini model also solved this issue as its vast training data provided the ability to recognize a much wider and more diverse range of car brands.

- **High Latency and API Costs from Continuous Video Analysis**

  The third challenges of the chosen client-server architecture was the potential for high latency and significant operational costs associated with making API calls to Gemini for every video frame. Continuously streaming video for analysis was not feasible. This was addressed by implementing a hybrid architecture. A fast, locally-run YOLO11n-seg model was integrated on the back-end to act as an intelligent pre-filter, analysing frames to detect and segment a car before any data was sent to the cloud. This ensured that an API call was only made for a single, high-quality image of a confirmed car, dramatically reducing both cost and processing time.

- **Client-Server Network Connectivity in a Development Environment**

  The last challenges was establishing a network connection between the physical iPhone client and the Python server running on a local development machine. Standard local networking does not allow external devices to connect directly. This was resolved by using ngrok, a reverse proxy service. Ngrok generated a secure, public URL that tunneled traffic directly to the local Flask server, creating a stable and reliable communication channel that was essential for rapid front-end development and testing.

5.6 Concluding Remark

This chapter detailed the full implementation of the TagT system. It outlined the hardware and software foundations, specified the critical configuration parameters that govern the system's behaviour and provided a step-by-step walkthrough of its real-world operation. The key challenges encountered during development and their solutions were also discussed. The outcome of this implementation phase is a robust, functional prototype of the ANPR application, which now forms the basis for the comprehensive system evaluation and performance analysis presented in Chapter 6.

CHAPTER 6 SYSTEM EVALUATION AND DISCUSSION

This chapter presents a comprehensive evaluation of the fully implemented TagT system. The primary objective is to quantitatively measure the performance of the final prototype, focusing on two key areas: the efficiency of its hybrid architecture and its recognition accuracy across a variety of real-world scenarios. The chapter details the performance metrics, testing setups, and final results, followed by a discussion of the project's challenges and an evaluation of how the final system met its initial objectives.

6.1 System Testing and Performance Metrics

To systematically evaluate the performance of the implemented TagT system, a set of specific performance metrics were defined. These metrics are divided into two categories to align with the two distinct evaluations performed in this chapter: Efficiency Metrics to assess the performance of the system's architecture, and Accuracy Metrics to assess the quality of its recognition results in real-world conditions.

6.1.1 Efficiency Metrics

These metrics were used in the Section 6.2.1 Architectural Efficiency Evaluation to compare the performance of the implemented hybrid architecture against a baseline approach.

- **Gemini API Time**

  It is measured in seconds and a specific duration of the API call to the Gemini model. This metric was chosen over total processing time to isolate and compare the performance of the most time-intensive step in both architectures.

- **API Cost**

  It is measured in USD and the direct monetary cost reported by the Gemini API for a single analysis, calculated based on the number of input and output tokens.

6.1.2 Accuracy Metrics

These granular metrics were used in the Section 6.2.3 Real-World Performance Evaluation to measure the success of the attribute recognition stage. For this evaluation, it is assumed that the YOLO model has successfully detected a car, as its high performance was validated in Chapter 4.

- **Overall Success Rate**

  This is the primary top-level metric. A detection attempt is classified as a "Success" only if all three attributes: License Plate Number, Car Brand and Colour are correctly identified in a single analysis. This measures the end-to-end reliability of the recognition pipeline.

- **Attribute-Specific Accuracy**

  To provide a more detailed breakdown of performance, the accuracy for each individual attribute was also calculated across all test cases within a scenario. This helps to identify which part of the recognition task is most challenging.

  - **Car Brand Accuracy**

    The percentage of tests where the car's brand was correctly identified.
  - **Car Colour Accuracy**

    The percentage of tests where the car's colour was correctly identified.
  - **License Plate Accuracy**

    The percentage of tests where the license plate number was correctly identified.

6.2 Testing Setup and Results

The evaluation was divided into two distinct tests. The first test was designed to validate the efficiency of the system's architecture, while the second was designed to assess its accuracy and robustness in real-world conditions.

6.2.1 Architectural Efficiency Evaluation – Testing Setup

This evaluation was designed to quantitatively justify the final hybrid architecture by measuring its efficiency gains in processing time and API cost against a baseline, "brute-force" approach.

This test was conducted using a dataset of nine pre-recorded videos that represent a range of typical driving and environmental conditions. Each video was processed using two distinct methodologies:

1. **Baseline Method (Screenshot and send frames to Gemini)**
   For this method, video frames were sampled at a high rate which is 25 FPS and sent sequentially to the Gemini API for analysis without any pre-filtering or resizing. This represents a simple but computationally expensive approach.

2. **Implemented Hybrid Method (Screenshot will be optimized by YOLO and ResNet before send to Gemini)**
   For this method, the same videos were processed by the final TagT system. The back-end uses a local YOLO11n-seg model to detect cars and a ResNet18 model with cosine similarity to filter out duplicate or near-identical frames. This intelligent pre-filtering pipeline selects only a small number of unique, relevant frames to be resized and sent to the Gemini API for analysis.

For each of the nine videos, two key performance metrics were measured for both methodologies:

1. **The Gemini API Time**
   Time taken for the Gemini API call to complete.
2. **The Total API Cost**
   Cost calculated from token usage while calling Gemini API.

6.2.2 Architectural Efficiency Evaluation - Results

The evaluation was conducted by processing a dataset of nine videos using both the Implemented Hybrid Method and the Baseline Method. The number of frames selected by each method for submission to the Gemini API was recorded and is presented in Table 6.2.2.1.

No.	Number of Frames send to Gemini		
	Hybrid	Baseline	Duration
Video1	29	278	9 Seconds
Video2	18	146	4 Seconds
Video3	10	252	8 Seconds
Video4	17	235	7 Seconds
Video5	26	458	15 Seconds
Video6	8	170	5 Seconds
Video7	31	303	10 Seconds
Video8	27	342	11 Seconds
Video9	8	112	3 Seconds

Table 6.2.2.1 Results of Number of Frames send to Gemini

The data in Table 6.2.2.1 reveals the efficacy of the Hybrid Method's intelligent pre-filtering pipeline. By utilizing YOLO for object detection and ResNet18 for similarity analysis, the Hybrid Method submitted a total of only 174 frames across all nine videos. In contrast, the Baseline Method, which sending frames at a high frequency, submitted 2296 frames. This represents a 92.4% reduction in the number of frames requiring analysis by the Gemini API.

Furthermore, this substantial reduction in submitted data has a direct and significant impact on both API processing time and cost, as visually represented in the comparative charts Figure 6.2.2.1.

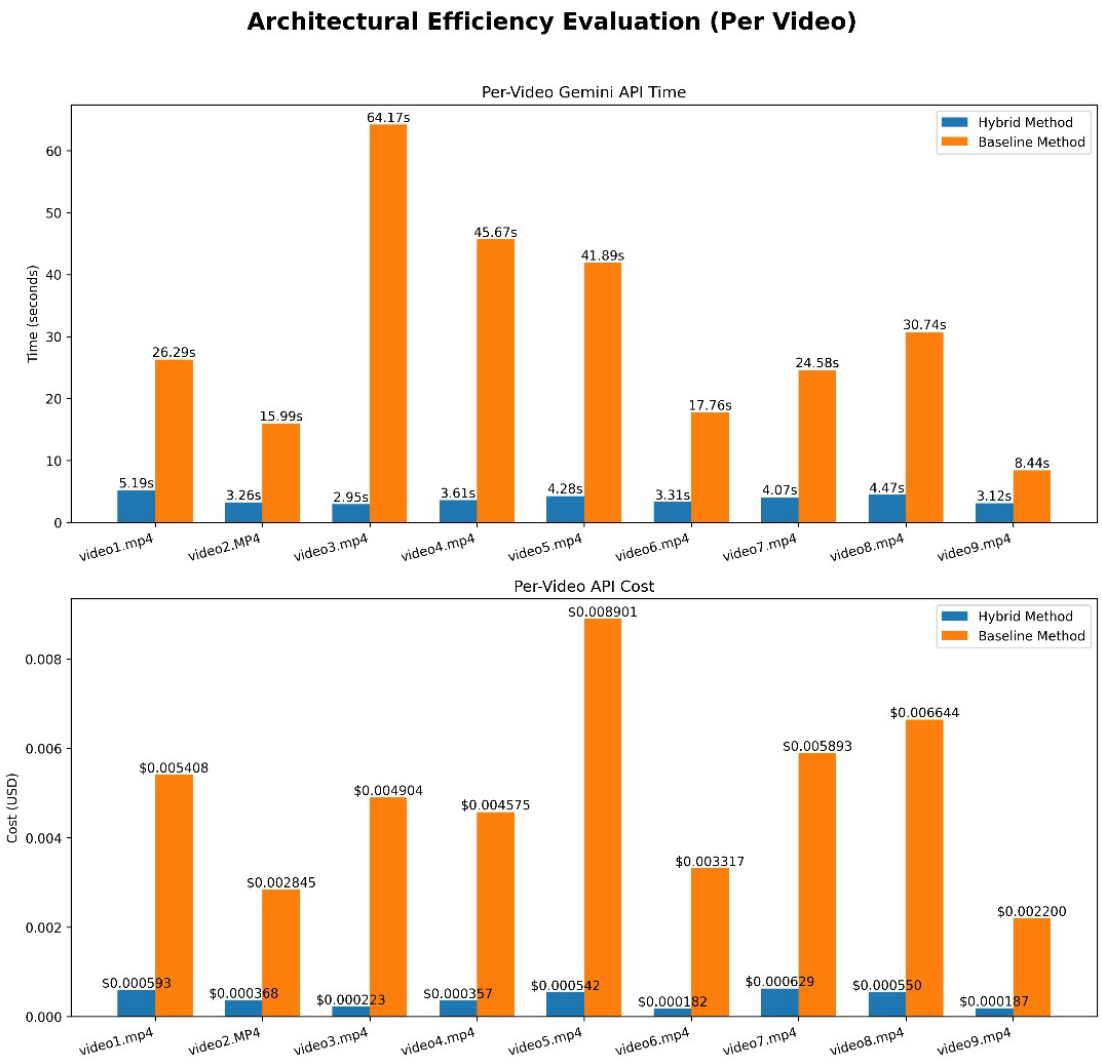


Figure 6.2.2.1 Charts of Comparison between Hybrid Method and Baseline Method

As shown in Figure 6.2.2.1, the Hybrid architecture yielded substantial performance gains:

- **Gemini API Time**

  The total time spent waiting for the Gemini API to process all requests was 34.26 seconds for the Hybrid Method, compared to 275.53 seconds for the Baseline Method. This constitutes a 87.57% reduction in processing latency, confirming that submitting fewer, smaller frames is significantly faster.

- **API Cost**

  The total operational cost for the Hybrid Method was $0.003631, while the Baseline Method was $0.044687. This represents a 91.87% reduction in API costs, directly attributable to the lower token count resulting from the reduced number of submitted frames.

To conclude, the architectural evaluation provides unequivocal, quantitative evidence that the implemented Hybrid Method is vastly superior to a baseline approach. By intelligently pre-filtering frames to reduce data volume by over 90%, the system achieves dramatic reductions in both processing time – 87.57% and operational cost – 91.87%. This confirms that the chosen architecture is a highly effective and optimized solution for a scalable, real-world application.

6.2.3 Real-World Performance Evaluation – Testing Setup

This evaluation was designed to assess the accuracy and robustness of the final, fully implemented TagT system across a range of real-world operational conditions.

This evaluation was conducted in the field using the final iOS application running on an iPhone 12 Pro. The system's performance was tested in real-time across eight distinct scenarios, which combined different environmental conditions: Daytime, Night-time, Rainy Day, Car Park Light and car viewing angles: Front and Side, as detailed in Table 6.2.3.1.

Scenario No.	Condition	Viewing Angle
1	Daytime	Front
2	Daytime	Side
3	Night-time	Front
4	Night-time	Side
5	Rainy Day	Front
6	Rainy Day	Side
7	Car Park Light	Front
8	Car Park Light	Side

Table 6.2.3.1 Eight Distinct Scenarios

To ensure a robust and reliable assessment of accuracy, a total of 20 detection attempts were made for each of the eight scenarios, resulting in 160 tests overall. For each attempt, a detection was marked as a "Success" only if the system correctly identified all three car attributes: License Plate Number, Car Brand and Colour. The final accuracy for each scenario was then calculated as the percentage of successful detections.

The diversity of the testing scenarios is illustrated in Figures 6.2.3.1 and Figure 6.2.3.2. These figures present a representative sample of the real-world detections, highlighting the system's performance across various vehicles and challenging conditions.

Day Time with Front Angle | Day Time with Side Angle

Night Time with Front Angle | Night Time with Side Angle

Figure 6.2.3.1 Day Time and Night Time Scenarios

Rainy Day with Front Angle | Rainy Day with Side Angle

Car Park Light with Front Angle | Car Park Light with Side Angle

Figure 6.2.3.2 Rainy Day and Car Park Light Scenarios

6.2.4 Real-World Performance Evaluation – Results

The overall performance of the TagT system across the eight real-world scenarios is summarized in Table 6.2.4.1. This table presents both the Overall Success Rate and the granular, Attribute-Specific Accuracy for each test case. A more detailed qualitative analysis for each environmental condition follows.

Condition	Viewing Angle	Overall Success Rate	License Plate Accuracy	Car Brand Accuracy	Car Colour Accuracy
Daytime	Front	90% (18/20)	95% (19/20)	100% (20/20)	95% (19/20)
	Side	85% (17/20)	95% (18/20)	100% (20/20)	100% (20/20)
Night-time	Front	80% (16/20)	95% (19/20)	95% (19/20)	90% (18/20)
	Side	80% (16/20)	85% (17/20)	100% (20/20)	95% (19/20)
Rainy Day	Front	90% (18/20)	100% (20/20)	95% (19/20)	95% (19/20)
	Side	85% (17/20)	85% (17/20)	100% (20/20)	100% (20/20)
Car Park Light	Front	85% (17/20)	90% (18/20)	100% (20/20)	95% (19/20)
	Side	75% (15/20)	85% (17/20)	95% (19/20)	95% (19/20)
Average	-	83.75%	91.25%	98.13%	95.63%

Table 6.2.4.1 Results of Eight Distinct Scenarios

The evaluation results, summarized in Table 6.2.4.1, provide a quantitative validation of the TagT system's performance. The system achieved a strong average Overall Success Rate of 83.75% across all 160 test cases. The attribute-specific data reveals that the system's core visual

recognition is exceptionally robust, while license plate reading remains the most significant challenge.

1. **Performance in Optimal Conditions (Daytime and Car Park)**

   In well-lit environments such as daytime and car park lighting, the system demonstrated excellent performance. The Car Brand Accuracy was consistently near-perfect, averaging 98.75% across these four scenarios. This indicates the Gemini model's powerful ability to recognize car brands from various angles. Besides, the Car Colour Accuracy was also very high, averaging 96.25%. The Overall Success Rate in these conditions averaged 83.75%, with failures almost exclusively linked to the License Plate Accuracy which is averaging 91.25%. This confirms that under good lighting, the system is highly reliable however its performance is primarily limited by the inherent challenges of reading license plate number from varied perspectives. Figure 6.2.4.1 has shown some images to represent sample of detections in well-lit conditions.

Figure 6.2.4.1 Representative Sample of Detections in Well-Lit Conditions

2. Performance in Challenging Conditions (Night-time and Rainy Day)

The system's robustness was further evaluated under more adverse conditions, where it continued to demonstrate impressive performance. The Car Brand Accuracy remained remarkably high, averaging 97.5% across these four challenging scenarios, showcasing the Gemini model's powerful resilience to poor lighting and environmental interference. Similarly, the Car Colour Accuracy was strong, averaging 95%. The License Plate Accuracy remained high at an average of 91.25%, confirming that the system can effectively handle challenges like low light and rain. The Overall Success Rate in these challenging conditions averaged 83.75%, which is identical to the performance in optimal conditions. This surprising and powerful result indicates that while individual attributes can be slightly affected by adverse conditions, the system maintains an extremely high level of operational reliability. Figure 6.2.4.2 has shown some images to represent sample of detections in challenging conditions.

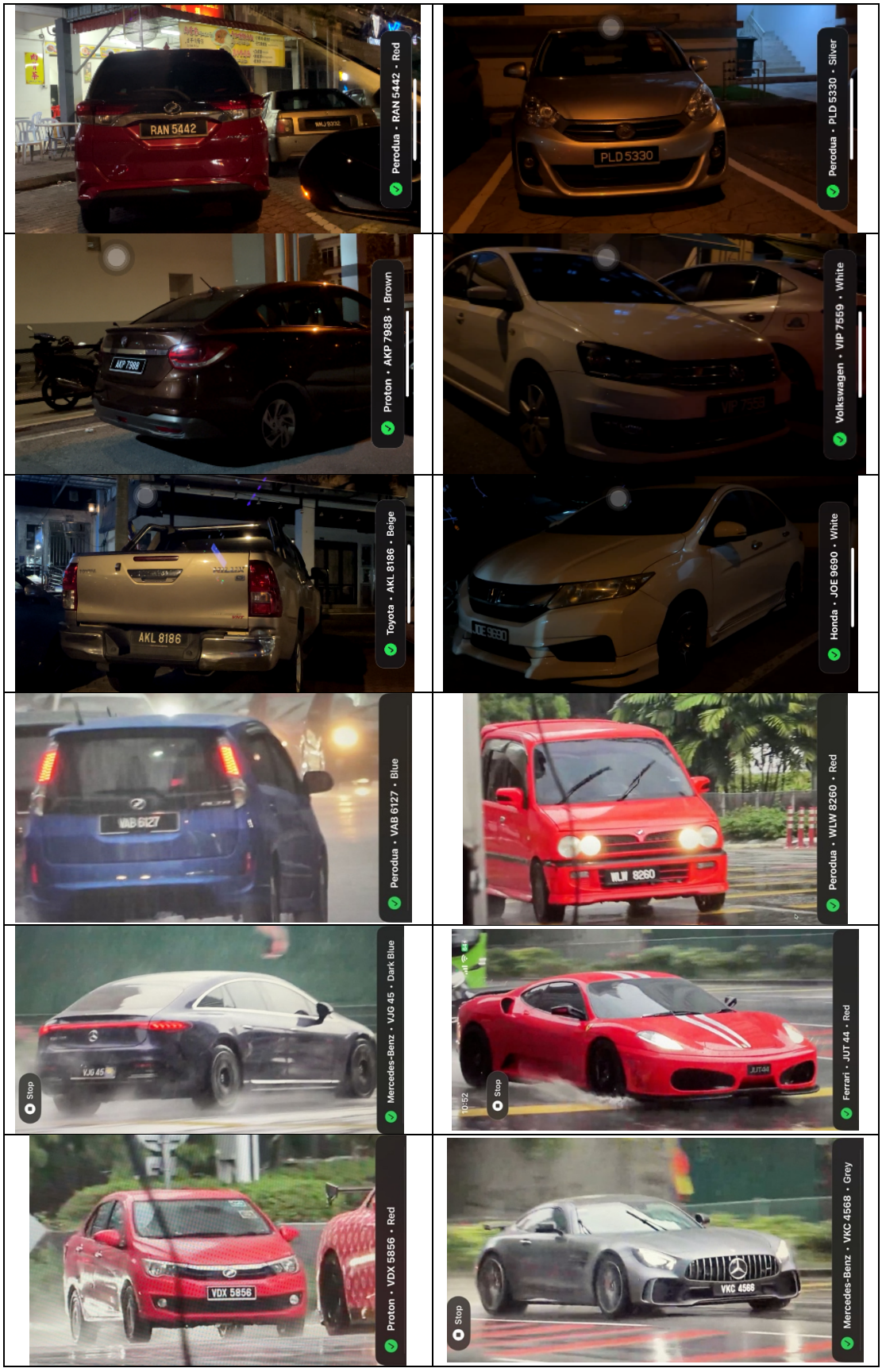


Figure 6.2.4.2 Representative Sample of Detections in Challenging Conditions

The data from the comprehensive real-world evaluation leads to two clear and significant conclusions regarding the performance of the TagT system.

1. **Core Visual Recognition is Extremely Robust**

   The system's ability to identify a car's brand and colour is its most powerful and reliable feature. The average Car Brand Accuracy of 98.13% is near-perfect, demonstrating the Gemini model's exceptional capability to recognize car brands from various angles and in diverse lighting conditions. Similarly, the Car Colour Accuracy of 95.63% is very high, showing only minor sensitivity to challenging lighting. This validates the choice of the Gemini model for its powerful, holistic visual analysis capabilities, which are highly resilient to environmental changes.

2. **Attribute Recognition Accuracy as the Primary Performance Driver**

   The Overall Success Rate, which requires all three attributes to be correct, serves as the strictest measure of end-to-end performance. The system achieved a strong average of 83.75% in this metric. This overall rate is primarily influenced by the performance of the two most challenging recognition tasks, which are License Plate Accuracy and Color Accuracy. While both accuracy percentages are high, failures in either of these categories contributed to the final success rate. The data indicates that License Plate Recognition is highly accurate, however, it is the most sensitive to environmental factors like viewing angle, lighting and physical obstructions. Therefore, it represents the most significant opportunity for future improvement.

In conclusion, the TagT system has been successfully validated as a highly effective proof-of-concept. It achieves its goal of multi-attribute car recognition with a high degree of success, achieving an average Overall Success Rate of 83.75% across 160 demanding real-world tests. The granular performance data not only proves the system's current capabilities but also provides a clear and data-driven path forward for future refinements.

6.3 Project Challenges

Throughout the development and evaluation of the TagT system, several key challenges were encountered that influenced the final design and highlighted important considerations for real-world deployment. Overcoming these challenges through iterative design and testing was crucial to achieving the project's objectives.

One of the primary challenges was the inherent unreliability of a traditional, multi-model ANPR pipeline. The initial design, which relied on separate models for object detection, OCR and each car attribute recognition, has been proved to be unfeasible. The OCR models for license plate reading performed very poorly and the specialized car brand recognition models lacked the scalability for real-world use. This fundamental design flaw was addressed by re-architecting the entire system using a Gemini model. This pivot completely bypassed the need for a fragile OCR step and provided the necessary flexibility for recognizing a wide variety of car brands.

A second major challenge was mitigating the high latency and operational costs associated with a cloud-based AI model. Sending a continuous video stream directly to the Gemini API was not a viable solution. This was overcome by designing a hybrid architecture. A lightweight YOLO11n-seg model was implemented on the back-end to act as an intelligent pre-filter. This module analyses frames locally to detect and segment cars first, ensuring that an API call is only made for a single, high-quality image of a confirmed target. This design dramatically reduced both API costs and overall processing time.

Finally, the system's performance was significantly impacted by environmental factors such as lighting and viewing angle. In low-light, night-time conditions, the colour of streetlights often caused inaccurate car colour classifications. Similarly, sharp viewing angles could distort license plate characters, leading to recognition errors. While no system can eliminate all environmental variability, this was mitigated by configuring the system to prioritize reliability by using a 640x480 resolution and by acknowledging in the final evaluation that the system performs optimally in frontal-view scenarios, which aligns with its primary intended use cases. The solutions and design choices applied in response to these challenges resulted in a more robust, efficient and cost-effective system.

6.4 Objectives Evaluation

This section reviews the outcomes of the TagT project in relation to the primary objectives established in Chapter 1. The comprehensive evaluation confirms that the project was successful in achieving its goals of enhancing ANPR security, designing an efficient and cost-effective architecture, and delivering a robust, user-friendly prototype. The final implemented system effectively addresses the technical, security and operational requirements outlined at the project's outset.

The core technical objective of enhancing ANPR security through multi-attribute recognition was fully achieved. The system successfully integrates a YOLO model for detection with a Gemini model for recognition, a combination proven to be highly effective. The Real-World Performance Evaluation demonstrated the system's ability to accurately identify not just a car's license plate number, but also its brand and colour with high fidelity, achieving an average Brand Accuracy of 98.13% and Colour Accuracy of 95.63%. By providing this multi-faceted data, the system directly fulfils its primary security goal of creating a tool to combat fraudulent activities like plate-swapping, which rely on the anonymity of single-point identification.

Furthermore, the project successfully delivered an architecture that is both efficient for real-time processing and cost-effective. The Architectural Efficiency Evaluation provided quantitative proof of this, showing that the implemented hybrid design is significantly faster and cheaper than a baseline approach, achieving a 87.57% reduction in API processing time and a 91.87% reduction in API costs. This result confirms that the system meets its objective of being a financially sustainable solution suitable for dynamic, real-world operational environments.

Finally, a robust and user-friendly prototype was successfully delivered and validated. The system's reliability was proven across eight varied and challenging environmental conditions, where it maintained a strong average Overall Success Rate of 83.75%. The final iOS application provides a seamless, intuitive and fully automatic user experience, as demonstrated in Chapter 5. This fulfils the crucial objective of creating a practical, proof-of-concept system that is both reliable in its performance and simple in its operation, successfully validating the project's goals.

CHAPTER 7 CONCLUSION AND RECOMMENDATION

7.1 Conclusion

In conclusion, the TagT project represents a significant and successful advancement in the field of Automatic Number Plate Recognition. It directly addresses the critical vulnerabilities of traditional ANPR systems which are their susceptibility to license plate fraud and the inflexibility of hardware-based solutions, by delivering a modern, software-based and car multi-attribute recognition framework. The project's success is founded on a rigorous, evidence-based methodology that systematically evaluated a wide range of technologies before arriving at an optimal hybrid architecture.

The final system architecture, which integrates YOLO11n for high-speed vehicle detection, ResNet18 with cosine similarity for intelligent frame optimization and the Gemini model for robust attribute recognition, has been proven to be both highly effective and remarkably efficient. The comprehensive evaluation in Chapter 6 provided quantitative proof of the design's superiority. The architectural efficiency tests demonstrated that by intelligently pre-filtering frames, the system achieved a 92.4% reduction in data sent for analysis, leading to a 87.57% decrease in API processing time and a 91.87% decrease in operational costs compared to a baseline approach.

Furthermore, the real-world performance evaluation confirmed the prototype's robustness and accuracy. The system achieved a strong average Overall Success Rate of 83.75% across 160 demanding tests in varied environmental conditions. The near-perfect Car Brand Accuracy - 98.13% validates the choice of a large-scale AI model, while the granular data highlights that the most significant remaining challenge is license plate reading from extreme angles. The final deliverable is a functional and intuitive iOS application where successfully meets all project objectives, establishing TagT as a powerful and validated proof-of-concept. By providing a replicable framework for building an efficient hybrid AI system, this project makes a meaningful contribution to the development of next-generation intelligent security solutions.

7.2 Recommendation

While the current TagT system successfully meets its core objectives as a robust proof-of-concept, several key enhancements could be explored to further improve its versatility, performance and applicability for global deployment.

One of the most significant areas for future work is the expansion and validation of the system with international license plates. The current prototype was validated primarily using Malaysian-style license plates. To evolve the system into a truly versatile and globally applicable solution, the next logical step would be to rigorously evaluate its performance against a diverse, multi-national dataset of license plates from regions such as Europe, North America and other parts of Asia. This would test the true extent of the Gemini model's capabilities in handling a wide variety of plate formats, fonts, character sets and syntaxes. Such an expansion would not only validate the system's scalability but also provide valuable insights into any regional biases in the model, guiding further refinements for a commercial-grade, international product.

Another powerful area for future enhancement lies in optimizing the system for on-device, offline deployment. The current client-server architecture, while highly effective, is fundamentally dependent on a stable internet connection, which limits its use in remote or low-connectivity areas. Future development could focus on exploring the feasibility of converting the current recognition pipeline into a lightweight version that can run directly on a mobile device's neural engine. This would involve researching and testing smaller, highly optimized models that could perform the recognition task locally. While this would likely introduce a trade-off in accuracy compared to the larger, cloud-based Gemini model, it would enable full offline functionality, eliminate network latency and remove all operational API costs. This would transform the TagT system into a completely self-contained tool, significantly increasing its portability and making it accessible for a much broader range of security and management applications.

REFERENCES

[1] FMT Reporters, “Foreigners who switch plates to buy petrol face jail, fine,” *Free Malaysia Today | FMT*, Nov. 2024. https://www.freemalaysiatoday.com/category/nation/2024/11/01/foreigners-who-switch-plates-to-buy-petrol-face-jail-fine/ (accessed Apr. 12, 2025).

[2] M. Y. Aalsalem, W. Z. Khan, and K. M. Dhabbah, “An automated vehicle parking monitoring and management system using ANPR cameras,” *IEEE Xplore*, Jul. 01, 2015. https://ieeexplore.ieee.org/abstract/document/7224887

[3] T. Mustafa and M. Karabatak, “Challenges in Automatic License Plate Recognition System Review,” *IEEE Xplore*, May 01, 2023. https://ieeexplore.ieee.org/document/10131688

[4] B Marisekar, T Abishek, M Dheeraj, R Adharsh, R. A. Raja, and A. U. Kaushik, “Smart Parking Fare Collection and Management System using ANPR Technology,” pp. 1699–1706, Sep. 2024, doi: https://doi.org/10.1109/icosec61587.2024.10722233.

[5] Samuel, “Kent driver found number plate was cloned after crash 280 miles away,” Mar. 14, 2025. Available: https://www.bbc.com/news/articles/cgj5w7wpg95o

[6] J. McLogan, “Fake New York license plates now have the complete attention of law enforcement,” *Cbsnews.com*, Feb. 13, 2025. https://www.cbsnews.com/newyork/news/new-york-state-fake-license-plates/

[7] “Number plate scams on the rise as thousands of drivers risk unfair Ulez fines,” *www.gbnews.com*. https://www.gbnews.com/lifestyle/cars/number-plate-scams-unfair-ulez-fines (accessed Jul. 15, 2024).

[8] P. Kanimozhi, S. Harshini, T. A. Kumar, and M. J. Mary, “Enhancing smart city security: IoT-driven ANPR technology for identifying smuggling vehicles,” *IET conference proceedings.*, vol. 2024, no. 37, pp. 366–371, Mar. 2025, doi: https://doi.org/10.1049/icp.2025.0940.

[9] A. Agarwal, S. Shinde, S. Mohite, and S. Jadhav, “Vehicle Characteristic Recognition by Appearance: Computer Vision Methods for Vehicle Make, Colour, and License Plate Classification,” *2022 IEEE Pune Section International Conference (PuneCon)*, pp. 1–6, Dec. 2022, doi: https://doi.org/10.1109/punecon55413.2022.10014731.

[10] Muhammad, Nor Azuana Ramli, and Mohd, "Car Logo Recognition using YOLOv8 and Microsoft Azure Custom Vision," pp. 477–481, Oct. 2023, doi: https://doi.org/10.1109/icdabi60145.2023.10629291.

[11] T. Mustafa and M. Karabatak, "Real Time Car Model and Plate Detection System by Using Deep Learning Architectures," *IEEE Access*, vol. 12, pp. 107616–107630, 2024, doi: https://doi.org/10.1109/access.2024.3430857.

[12] T. Kim, C.-H. Kang, Y. Kim, and S. Yang, "AI Camera: Real-time License Plate Number Recognition on Device," *2022 IEEE International Conference on Consumer Electronics (ICCE)*, Jan. 2022, doi: https://doi.org/10.1109/icce53296.2022.9730306.

[13] Pankaj Mukhija, Pawan Kumar Dahiya, and Priyanka, "Challenges in Automatic License Plate Recognition System: An Indian Scenario," Jul. 2021, doi: https://doi.org/10.1109/ccict53244.2021.00055.

[14] Z. Rahman, A. M. Ami, and M. A. Ullah, "A Real-Time Wrong-Way Vehicle Detection Based on YOLO and Centroid Tracking," *IEEE Xplore*, Jun. 01, 2020. https://ieeexplore.ieee.org/document/9230463 (accessed Apr. 06, 2023).

[15] Md. Rahatul Islam and K. Horio, "Real Time-Based Face Recognition, Tracking, Counting, and Calculation of Spent Time of Person Using OpenCV and Centroid Tracker Algorithms," Jun. 2023, doi: https://doi.org/10.1109/iccci59363.2023.10210102.

[16] A. Anish, S. R, A. Hema Malini, and T Archana, "Enhancing Surveillance Systems with YOLO Algorithm for Real-Time Object Detection and Tracking," Dec. 2023, doi: https://doi.org/10.1109/icacrs58579.2023.10404710.

[17] L. N. Rani and Yuhandri Yuhandri, "Similarity Measurement on Logo Image Using CBIR (Content Base Image Retrieval) and CNN ResNet-18 Architecture," Feb. 2023, doi: https://doi.org/10.1109/iccosite57641.2023.10127711.

[18] Muhammed Murat Özbek and Hilal Tekgöz, "Image Retrieval with Segment Anything and CNN Techniques," *2021 6th International Conference on Computer Science and Engineering (UBMK)*, pp. 131–136, Sep. 2023, doi: https://doi.org/10.1109/ubmk59864.2023.10286583.

[19] C. Hu, X. Bai, L. Qi, X. Wang, G. Xue, and L. Mei, "Learning Discriminative Pattern for Real-Time Car Brand Recognition," vol. 16, no. 6, pp. 3170–3181, Jun. 2015, doi: https://doi.org/10.1109/tits.2015.2441051.

[20] Z. Tong, Y. Wang, Y. Xue, Z. Zeng, Y. Ding, and Y. Ning, "Vehicle Colour Recognition Algorithm Based on Colour Space and Sector Features and Related Laws and Regulations," *2022 2nd International Conference on Consumer Electronics and Computer Engineering (ICCECE)*, pp. 765–768, Jan. 2022, doi: https://doi.org/10.1109/iccece54139.2022.9712816.

[21] S. Ghanem and J. H. Holliman, "Impact of Colour Space and Colour Resolution on Vehicle Recognition Models," *Journal of Imaging*, vol. 10, no. 7, p. 155, Jun. 2024, doi: https://doi.org/10.3390/jimaging10070155.

APPENDIX

Poster

Tan Jo Fang
23ACB00641

Universiti Tunku Abdul Rahman, Faculty of Information and Communication Technology
FINAL YEAR PROJECT JUNE 2025

TagT: A Versatile ANPR Solution for Diverse Applications

Introduction

TagT is an advanced Automatic Number Plate Recognition (ANPR) system designed to combat license plate fraud by identifying multiple car attributes:

- FYP 2 License Plate Numbers
- Car Brands
- Car Colours

Problems

1 License Plate Fraud

Foreign drivers swapped plates to Malaysia plates to access subsidized RON95 petrol.

2 Rigidity of Hardware-based ANPR Systems

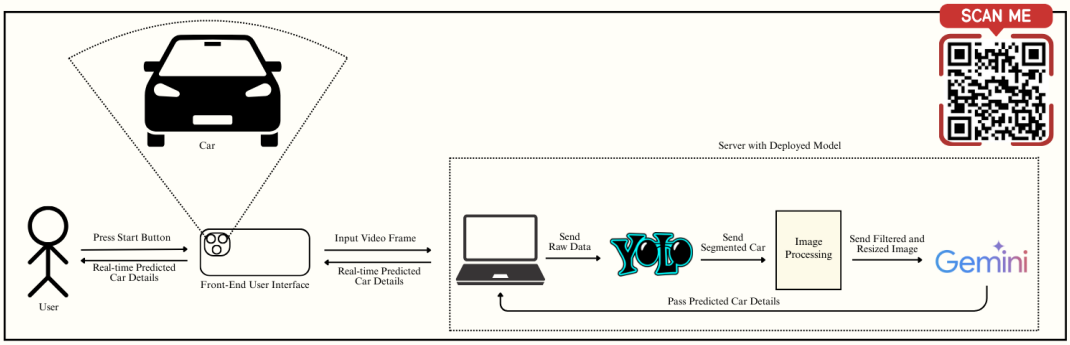
Most hardware-based ANPR systems are hard to update because their software is fixed in specialized devices and this limits to add new algorithms.

Deliverables

Successfully delivers a **robust, cost-effective and validated proof-of-concept** for a **modern ANPR system**.

By integrating lightweight local models with advanced AI models, this project provides a replicable framework that significantly enhances security and overcomes the limitations of traditional systems.

Proposed Solution - Real Time Car Detection and Multi-Attribute Recognition



Architectural Efficiency Evaluation

- Frame Numbers send to Gemini **92.4%** ↓
- Gemini Processing Time **87.6%** ↓
- Gemini API Cost **91.9%** ↓

Real-World Performance Evaluation

Average Overall Success Rate **83.8%**

Car Brand Accuracy **98.1%**

Car Colour Accuracy **95.6%**

License Plate Number Accuracy **91.3%**

160 images tested in Eight different scenarios

Contributions

For Researchers:

Presents a head-to-head test of Roboflow models and advanced AI models. It shows their actual performance on a real task, comparing their speed, cost and accuracy to help others choose the best tool.

For Developers:

Showing a modern ANPR framework to enhance car identification. This project demonstrates using Gemini to recognize car's license plate number, car brand and colour at the same time.

For Businesses:

Proves a cost-saving hybrid ANPR architecture. Using YOLO and ResNet18 as a pre-filter can reduce API cost and processing time by over 90%. This establish a financially sustainable for developing real-time AI applications.