No d'ordre: 07/2026-D/IN

République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université des Sciences et de la Technologie Houari Boumediene

Faculté d'Informatique

Thèse de Doctorat en Science

Présentée pour l'obtention du grade de Docteur

En : *Informatique*

Spécialité : Informatique

Par : **Ali KHALFI**

Thème :

Surveillance intelligente basée sur la recherche d'image dans les villes intelligentes

Soutenue publiquement, le 15 / 01 /2026 devant le jury composé de :

M. LARABI Slimane	Professeur à l'USTHB	Président
M. GUERROUMI Mohamed	Professeur à l'USTHB	Directeur de thèse
Mme. BOUGHACI Dalila	Professeur à l'USTHB	Examinatrice
M. TAHAR ABBES Mounir	Professeur à U.H.B Chlef	Examinateur
M. HADJILA Fethallah	Maitre de conférence /A à U.A.B.B Tlemcen	Examinateur
M. HACHAMA Mohammed	Professeur à E.S.M Sidi Abdellah	Examinateur

Acknowledgements

I want to thank my thesis supervisor, Prof. GUERROUMI Mohamed, for coming up with the thesis topic, for his help, his advice, and all the work he put into finishing this project. I also praise him for being a good person, being there for me, being humble, encouraging me, and understanding me. Every time I saw him, our talks made me feel better and pushed me to keep working. I hope he finds here all of my respect, appreciation, and thanks.

I would also want to thank Prof. LARABI Slimane for being the chair of the committee that reviewed my work. I would also want to thank Prof. HACHAMA Mohammed, Doctor. HADJILA Fethallah, Prof. TAHAR ABBES Mounir and Prof. BOUGHACI Dalila, for the honor of consenting to help me evaluate this study.

I also want to thank my mother from the bottom of my heart for her unconditional love and support throughout my life. A big thank you to my wife for always being there for me and cheering me on. To my children, Fatma zohra, Safaa, Roquia, Mohamed and Ahmed. Your pride in my work has been a huge source of motivation for me. I also ask for their understanding for the many hours this thesis took, which I wish I could have spent with them instead.

Finally, I want to thank everyone who has helped me with this project, either directly or indirectly. I am especially thankful to my colleagues at the University of Khemis Miliana, with whom I had many conversations, helpful exchanges, and support as we worked on our doctoral theses at the same time. To those who are still working on their theses or habilitation, I hope you finish quickly and successfully, Inshallah.

May Allah forgive my father's sins, have mercy on his soul, and grant him the highest level of Paradise. May Allah grant my father Jannatul Firdaus.

Abstract

Significant financial and environmental costs have resulted from the escalation of issues with parking inefficiency and traffic congestion brought on by the fast urban population growth. In order to solve these problems, Intelligent Transportation Systems (ITS) are essential, and visual data analysis has emerged as a viable and affordable option. By addressing two related issues parking space occupancy detection and road traffic congestion classification, this doctoral thesis suggests innovative artificial intelligence models based on image analysis to improve intelligent surveillance in smart cities.

Three significant additions to intelligent transportation systems are suggested by the thesis. It starts by creating a parking management plan based on deep learning. Spaces can be classified as occupied or vacant with high accuracy and efficiency using a customized CNN with dilated convolutions, and generalization is further enhanced by a transfer learning strategy that uses Inception V3 and data augmentation. The PKLot dataset is used to validate both. Second, it presents a new framework for traffic congestion that uses single-head attention in conjunction with a lightweight Vision Transformer (UniViT) and YOLOv8s for vehicle detection. This model outperforms current techniques by classifying congestion levels (light, medium, and heavy) with an accuracy of 99.89% on the UCSD dataset. Third, it introduces a Multi-Module Road Traffic Estimation (MRTE) system that estimates density and speed using SIFT method, avoiding vehicle tracking. Congestion levels are then determined by a fuzzy logic inference system, which achieves 99.47% accuracy with low computational cost and high interpretability.

In general, the thesis offers precise, effective, and useful solutions for managing urban mobility. It provides real-time tools to improve parking, ease traffic, and aid in the creation of smart and sustainable cities.

Keywords: Smart Cities, Intelligent Transportation Systems (ITS), Deep Learning, Convolutional Neural Networks (CNN), Vision Transformer (ViT), YOLO, Parking Occupancy Detection, Traffic Congestion Classification, Fuzzy Logic, Image Processing.

Résumé

Des coûts financiers et environnementaux significatifs ont résulté de l'escalade des problèmes d'inefficacité du stationnement et de congestion du trafic causés par la croissance rapide de la population urbaine. Afin de résoudre ces problèmes, les Systèmes de Transport Intelligents (STI) sont essentiels, et l'analyse de données visuelles est apparue comme une option viable et abordable. En abordant deux problèmes connexes, la détection de l'occupation des places de stationnement et la classification de la congestion du trafic routier, cette thèse de doctorat propose des modèles d'intelligence artificielle innovants basés sur l'analyse d'images pour améliorer la surveillance intelligente dans les villes intelligentes.

Trois ajouts significatifs aux systèmes de transport intelligents sont suggérés par la thèse. Cela commence par la création d'un plan de gestion du stationnement basé sur l'apprentissage profond. Les espaces peuvent être classifiés comme occupés ou vacants avec une grande précision et efficacité en utilisant un CNN personnalisé avec des convolutions dilatées, et la généralisation est encore améliorée par une stratégie d'apprentissage par transfert qui utilise Inception V3 et l'augmentation des données. Le dataset PKLot est utilisé pour valider les deux. Deuxièmement, il présente un nouveau cadre pour la congestion du trafic qui utilise l'attention à tête unique en conjonction avec un Vision Transformer léger (UniViT) et YOLOv8s pour la détection des véhicules. Ce modèle surpasse les techniques actuelles en classifiant les niveaux de congestion (léger, moyen et lourd) avec une précision de 99,89 % sur le jeu de données UCSD. Troisièmement, il introduit un système d'estimation du trafic routier multi-modules (MRTE) qui estime la densité et la vitesse en utilisant la méthode SIFT, évitant ainsi le suivi des véhicules. Les niveaux de congestion sont ensuite déterminés par un système d'inférence logique floue, qui atteint une précision de 99,47 % avec un faible coût de calcul et une grande interprétabilité.

En général, la thèse offre des solutions précises, efficaces et utiles pour gérer la mobilité urbaine. Elle fournit des outils en temps réel pour améliorer le stationnement, fluidifier le trafic et aider à la création de villes intelligentes et durables.

Mots-clés : Villes intelligentes, Systèmes de transport intelligents (STI), Apprentissage profond, Réseaux neuronaux convolutifs (CNN), Vision Transformer (ViT), YOLO, Détection de l'occupation des parkings, Classification de la congestion du trafic, Logique floue, Traitement d'images.

ملخص

تُؤدي مشكلات عدم كفاءة مواقف السيارات وازدحام المرور المتفاقمة، والناجمة عن النمو السكاني الحضري السريع، إلى تكاليف مالية وبيئية باهظة. ولمعالجة هذه المشكلات، تُعدّ أنظمة النقل الذكية ضرورية لذالك، وقد أثبت تحليل البيانات المرئية جدواه وفعاليته من حيث التكلفة. و ذلك من خلال معالجة مشكلتين مترابطتين - وهما رصد الاماكن المحجوزة و الفارغة عبرمواقف السيارات وتصنيف ازدحام المرور - تقترح هذه الأطروحة نماذج ذكاء اصطناعي مبتكرة، تعتمد على تحليل الصور، لتحسين المراقبة الذكية في المدن الذكية.

تقدم هذه الأطروحة ثلاثة إسهامات رئيسية لأنظمة النقل الذكية. أولًا، تُطوّر خطة لإدارة مواقف السيارات تعتمد على اقتراح و تصميم نموذجين يعتمدان على التعلّم العميق. حيث يُمكن تصنيف مواقف السيارات إلى مشغولة أو فارغة بدقة وكفاءة عاليتين و ذلك باستخدام شبكة عصبية التفافية مُخصصة ذات التفافات مُوسّعة. ثم يتم تحسين التعميم من خلال استراتيجية التعلّم بالنقل باستخدام Inception V3 ورفع حجم البيانات. تُستخدم مجموعة بيانات PKLot للتحقق من صحة هذين النموذجين. ثانيًا، يقدم هذا البحث إطارًا جديدًا لإدارة الازدحام المروري، حيث يستخدم آلية الانتباه أحادي الرأس مع مُحوِّل الرؤية الخفيف (UniViT) وخوارزمية YOLOv8s للكشف عن المركبات. يتفوق هذا النموذج على التقنيات الحالية بتصنيفه لمستويات الازدحام (منخفض، متوسط، مرتفع) بدقة تصل إلى 99.89% و ذلك باستعمال مجموعة بيانات جامعة كاليفورنيا في سان دييغو. كما يقدم نظامًا لتقدير حركة المرور متعدد الوحدات (MRTE) حيث يُقدِّر الكثافة والسرعة باستخدام طريقة SIFT، متجنبًا بذلك تتبع المركبات. ثم تُحدَّد مستويات الازدحام بواسطة نظام المنطق الضبابي، محققًا دقة تصل إلى 99.47% بتكلفة حسابية منخفضة وقابلية تفسير عالية.

بشكل عام، تقترح هذه الأطروحة حلولًا دقيقة وفعالة ومفيدة لإدارة التنقل الحضري. وتوفر حلول ذكية و فورية لتحسين مواقف السيارات، وتبسيط حركة المرور، والمساهمة في بناء مدن ذكية ومستدامة.

الكلمات المفتاحية: المدن الذكية، أنظمة النقل الذكية (ITS)، التعلم العميق، الشبكات العصبية الالتفافية (CNN)، محول الرؤية (ViT)، YOLO، كشف إشغال مواقف السيارات، تصنيف الازدحام المروري، المنطق الضبابي، معالجة الصور.

Contents

List of Figures

List of Tables

Chapter 1: Research context and motivation of the study

1.1 Introduction and context

Due to the world's growing urbanization, with 68% of people expected to live in cities by 2050, the concept of smart cities has become increasingly important [1]. Smart cities use modern technologies such as artificial intelligence (AI), the Internet of Things (IoT), and big data to create urban environments that are liveable, sustainable, and efficient. Within this context, intelligent transportation systems (ITS) play a key role, aiming to improve safety, reduce environmental impact, and optimize mobility. Traffic congestion and parking shortages remain major urban transport challenges, strongly affecting both the economy and quality of life. ITS helps address these problems by combining advanced sensing, communication, and computing technologies. According to [2], traffic congestion costs billions of dollars each year. In 2023 alone, delays, wasted fuel, and pollution caused losses of over $300 billion worldwide. In addition, drivers in large cities are estimated to spend 17–20% of their travel time searching for parking [3]. These issues highlight the need for ITS to provide data-driven, real-time solutions for better city transport management.

Traditionally, image retrieval and classification have been treated as different tasks. Image classification means assigning a label to an image, such as identifying whether a traffic scene is congested or clear, or whether a parking spot is occupied or free. Image retrieval involves searching through a large set of images to find those that are most relevant to a query image—for example, retrieving all images of a specific vehicle or finding visually similar traffic events.

However, research, including the work of [4], has shown that image classification and retrieval are closely related. Both can be solved by measuring the similarity between images or between an image and a set of category images. This shared approach has important benefits for smart city surveillance. For example, the same deep learning features and similarity measures used to classify parking occupancy can also be applied to find cases of illegal parking or to track vehicles across different cameras. In the same way, congestion detection can use both scene classification and the retrieval of similar past congestion patterns, supporting more advanced analysis and decision-making.

This thesis aims to develop intelligent monitoring solutions for smart cities. It focuses on detecting parking availability and road traffic congestion. The goal is to design accurate models

using artificial intelligence, especially deep learning, to achieve reliable and adaptable image classification for Intelligent Transportation Systems (ITS). The work also makes use of the shared foundations between image classification and image retrieval to improve performance.

1.2 Problem statement

The growth of smart city technologies has led to a massive increase in visual data from surveillance cameras, traffic monitors, and sensor networks. Using this data effectively for urban management tasks, such as monitoring parking spaces, detecting traffic congestion, and enabling intelligent surveillance, requires strong computer vision methods. These tasks fall within the broader framework of Intelligent Transportation Systems (ITS), a key element of smart cities.

Traditional approaches to parking and traffic control rely on costly and rigid infrastructure, such as magnetic sensors or inductive loops. These systems are expensive to install and maintain, limited in coverage, and unable to adapt to the evolving needs of modern cities. Manual supervision, another conventional method, is also inefficient because it is time-consuming and prone to human error.

Video-based surveillance using the existing camera infrastructure, offers a more flexible and scalable alternative for monitoring parking and traffic. However, it comes with several challenges:

- Environmental variability: Urban scenes change with lighting conditions (day/night cycles, shadows), weather effects (rain, fog, snow), and occlusions caused by vehicles or trees. These variations reduce image quality and make it harder to detect and classify parking occupancy or traffic congestion accurately.
- Performance of classification models: Inaccurate image classification models, even when built using advanced techniques like deep learning, can hinder the reliable detection of parking and congestion. This negatively impacts traffic management and weakens the effectiveness of ITS.
- Efficiency of image processing: Preparing images for training or classification is complex. Achieving high accuracy while keeping memory usage low is challenging, as reducing image size often sacrifices important features needed for reliable classification.

Parking and congestion are also closely linked. Poor parking management forces drivers to spend significant time searching for spaces, which increases both emissions and traffic congestion. In turn, congested roads reduce access to parking facilities, worsening urban

mobility. This cycle highlights the urgent need for integrated solutions that can tackle parking and congestion together.

The combined challenges of environmental variability, model performance, and efficient image analysis highlight the need for intelligent, integrated, and scalable surveillance systems within ITS. Addressing these problems is critical for sustainable urban mobility and for improving the quality of life in smart cities.

1.3 Thesis objectives

The main objective of this thesis is to design and develop intelligent solutions for road traffic surveillance by combining image analysis and artificial intelligence (AI) techniques. The work focuses on building effective models for image classification in smart city environments to improve Intelligent Transportation Systems (ITS). These models aim to support better parking management and traffic congestion detection by making use of visual data collected from surveillance and monitoring systems.

As shown in Figure 1.1, the research plan is divided into two main parts:

1. **Parking occupancy detection**: determining whether parking spaces are available or occupied using a customized Convolutional Neural Network (CNN) and transfer learning methods.
2. **Traffic congestion detection**: analysing traffic flow using a Vision Transformer model combined with fuzzy logic to assess road congestion levels.

Traffic congestion directly affects parking turnover, while the search for parking contributes to congestion. This feedback loop illustrates how parking and congestion are interconnected challenges in urban mobility. By addressing them together, this research proposes a holistic framework that improves mobility, reduces unnecessary traffic, and supports sustainable and intelligent surveillance in smart cities.

This work seeks to achieve high accuracy and performance in recognizing urban scenarios under practical conditions. The specific objectives of the research are detailed in the subsequent sections.

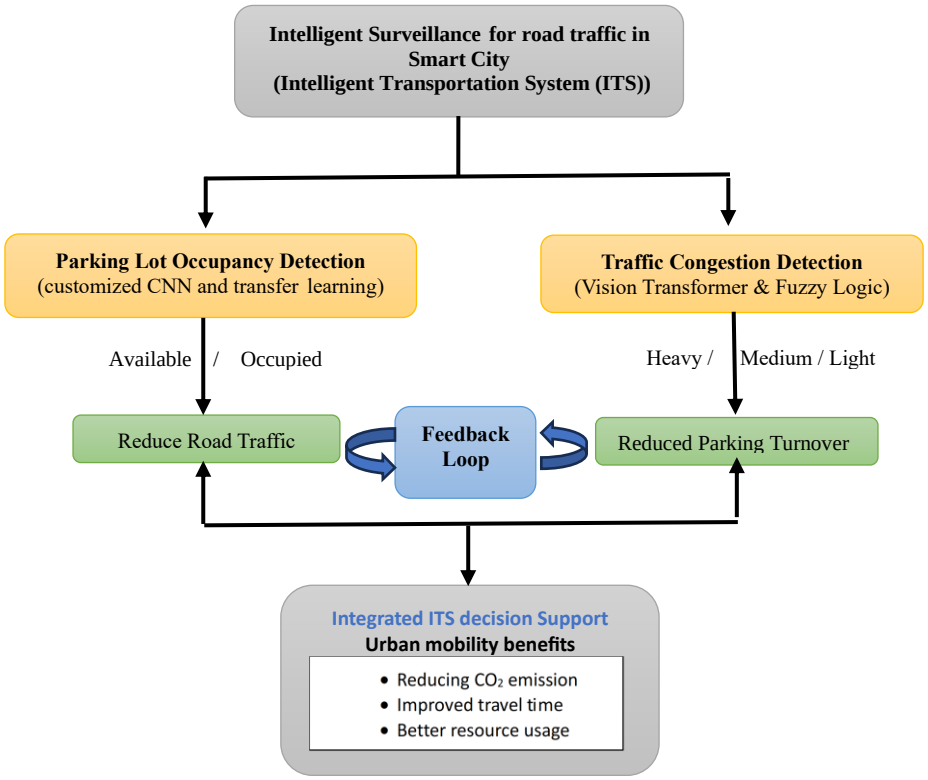


Figure 1.1: Parking and congestion are cyclically related in the context of smart city intelligent surveillance.

1.3.1 Parking occupancy detection objectives

The first goal is to create deep learning models for precise parking occupancy status identification. This includes creating CNN-based architectures to categorize roadside parking spots as "occupied" or "free," such as models with dilated convolutions. Computational efficiency is given a priority in order to facilitate deployment on low-power embedded systems. Additionally, the study uses the InceptionV3 architecture and transfer learning to improve categorization in a variety of environmental circumstances by utilizing pre-trained information. Techniques for augmenting picture data are used to improve generalizability. The PKLot dataset is used to train and evaluate these models, which are meant to serve as the basis for real-time smart parking management systems.

1.3.2 Traffic congestion detection objectives

A deep learning architecture based on vision is used to develop a system for categorizing the degree of urban traffic congestion. The Region of Interest (ROI) approach is used first, and then YOLOv8s image mapping is used for vehicle recognition. The lightweight Vision Transformer (UniViT) at the heart of the system uses single-head self-attention processes to categorize situations into three groups based on the level of congestion: light, medium, and

heavy. The UCSD dataset is used to evaluate the system's performance in order to guarantee dependability under a variety of circumstances. Real-time congestion detection is made possible by this vision-driven pipeline, which is an essential component of ITS-based surveillance.

In order to supplement deep learning models, this thesis presents a Multi-Module Road Traffic Estimation (MRTE) system that is driven by fuzzy logic. The method calculates congestion levels by using basic traffic parameters, such as vehicle speed and density derived via image processing. A fuzzy rule framework based on traffic flow theory and expert knowledge is used to assess traffic flow in real time. MRTE provides a versatile and interpretable method of assessing traffic conditions by integrating several analytical modules. It provides useful insights to traffic management authorities, allowing them to make well-informed judgments using both visually represented data and physically generated congestion indications.

1.4 Context and motivation

Urban infrastructure is under unprecedented strain due to the exponential rise of the population, especially in the transportation sector. Cities are depending more and more on Intelligent Transportation Systems (ITS) to handle the intricacies of urban transportation as they develop into smart cities. The use of visual data from sensors and security cameras to track traffic patterns and parking availability is a crucial component of this change [5]. But gathering visual data alone is insufficient. In order to facilitate both long-term planning and real-time decision-making, it is crucial that this data be intelligently interpreted [6].

Intelligent Transportation Systems (ITS) make heavy use of image classification, which gives images semantic labels in order to study and interpret complicated urban surroundings. For example, it is essential for figuring out whether parking spots are available or occupied, which helps with effective parking management and reduces traffic in cities. Similarly, determining whether traffic is congested or free-flowing allows for real-time traffic monitoring and dynamic control strategies to improve traffic flow and reduce financial losses. According to [7], sophisticated machine learning techniques facilitate automated congestion monitoring, allowing adaptive signal control and route guiding systems to minimize delays.

Advances in deep learning, particularly convolutional neural networks (CNNs), have dramatically enhanced the accuracy and robustness of image classification models, allowing them to handle diverse challenges such as varying lighting conditions, weather effects, and occlusions common in urban environments [8]. The ability to accurately classify images in real

time is thus essential for enabling intelligent surveillance, traffic management, and other ITS applications that contribute to the development of smarter and more sustainable cities.

For instance, a model trained to classify parking spaces can also support retrieval of similar occupancy patterns across multiple locations. Similarly, a traffic congestion classifier using image processing can be extended to retrieve archived congestion episodes for comparative analysis and forecasting.

In light of these advancements, the goal of this thesis is to expand on the widely accepted principles of picture classification by utilizing cutting-edge deep learning methods to facilitate intelligent surveillance in smart cities. This study tackles two closely related issues that have a big influence on urban mobility and sustainability: parking occupancy and traffic congestion monitoring. The ultimate purpose is to provide integrated, scalable solutions that support the more general goals of intelligent urban planning, effective traffic management, and real-time monitoring.

1.5 Challenges

Despite encouraging outcomes in every experimental component of this study, there are a number of practical and technical obstacles to overcome when creating classifier models for urban settings. These difficulties cover everything from preparation and data collection to real-time performance and model generalization. The main challenges faced during the development and assessment of the suggested models are listed below.

1.5.1 Image variability and environmental Factors

Urban visual scenes captured by roadside and aerial cameras are subject to significant variability due to changing lighting conditions (e.g., day/night cycles, shadows), weather effects (e.g., rain, fog, overcast), and partial occlusions (e.g., other vehicles, trees, infrastructure). These variations affect image quality and reduce the consistency of feature representation, especially in parking images and traffic scenes. During training and evaluation, such conditions resulted in performance degradation unless addressed with data augmentation or region-of-interest filtering.

1.5.2 Model overfitting and generalization

Training deep learning models on datasets such as PKLot or UCSD introduces the risk of overfitting, particularly when models are exposed to limited or imbalanced data subsets. For example, some parking areas in the PKLot dataset had clear backgrounds with minimal distractions (e.g., PUCPR), while others (e.g., UFPR04) contained complex scenes with obstacles, shadows, and irregular terrain. Similarly, traffic congestion levels in UCSD varied in

visibility due to rain, camera glare, or poor resolution. Generalizing models across these variations required careful tuning of hyperparameters, dropout, and augmentation strategies to maintain performance across different subsets.

1.5.3 Computational constraints and optimization

Creating models that were both computationally efficient and worked well for deployment on edge devices or smart cameras was one of the main challenges. Managing high parameter counts and memory use was the trade-off for the lower training costs that transfer learning with InceptionV3 offered. The accuracy of the Vision Transformer (UniViT) model was high (up to 99.89%), however transformer-based models are heavier than CNNs by nature and need attention methods, positional encoding, and patch embedding, which might slow down inference if not correctly tuned.

1.5.4 Lack of labeled data for prediction

While classification datasets like PKLot and UCSD offer labeled snapshots of parking occupancy or traffic states, forecasting future congestion requires additional time-series data or synthesized label sequences. The fuzzy logic–based MRTE system addressed this to some extent by combining real-time image-based speed and density estimations with a rule-based congestion evaluator. However, the lack of benchmark datasets for congestion forecasting using image input limited our ability to compare forecasting models against existing methods in a standardized way.

1.5.5 Balancing classification accuracy and memory efficiency

In real-world ITS applications, especially when deployed on embedded devices, maintaining high classification accuracy while reducing memory consumption and processing time remains a significant challenge. This was particularly evident in the ViT-based traffic classifier and in the multi-module fuzzy system, where pre-processing steps such as image mapping, ROI extraction, and vehicle detection introduce latency. Techniques like patch reduction, global pooling, and layer freezing were used to strike this balance, though continuous tuning was required to reach optimal configurations.

1.6 Contributions

This research contributes to the development of image classifier models both detection parking lot and traffic congestion for intelligent surveillance systems in smart cities in the following ways:

- Deep Learning Classification of Roadside Parking Spaces: An efficient system for detecting roadside parking occupancy using convolutional neural networks (CNN) is

proposed [168]. The model minimizes calculation parameters by employing specific layer parameterization and a progressive dilation rate, achieving robustness against disturbances like occlusions and shadows. Validated on the PKlot dataset, the solution is suitable for embedded environments due to its low computational resource requirements.

- Transfer Learning for Parking Occupancy Detection: explores the use of deep learning, specifically transfer learning with the Inception V3 model and image data augmentation, to classify parking spaces in the PKLot dataset [169]. The experimental results demonstrate promising accuracy and efficient computational performance.
- Vision Transformer for Traffic Congestion Detection: introduces a novel approach [170] for traffic video analysis using image mapping and the UniVit (Unit Vision Transformer) model. Video frames are processed with YOLOv8s for vehicle detection and converted into mapped images, which are then analyzed by the vision transformer. Evaluated on the UCSD dataset under various weather conditions, the method achieves a high accuracy of 99.89%, outperforming previous techniques. The results highlight the potential of combining image mapping with vision transformers for intelligent traffic management.
- Fuzzy Logic Multi-Module for Real-Time Traffic Congestion: A novel system [171] for real-time road traffic congestion prediction. It utilizes the SIFT algorithm for congestion analysis without the need to track individual vehicles, and applies generalized fuzzy control rules to handle various traffic scenarios. The system employs SIFT-based image processing to detect vehicle density and speed, treating vehicle tracking as non-essential for effective traffic detection. Fuzzy logic is used to provide comprehensive traffic information, demonstrating robust performance across different weather conditions using the UCSD dataset.

1.7 Thesis outline

The remaining content of this thesis is organized into the following chapters.

- **Chapter 2** presents the background on image classification, image retrieval, and related work. It introduces the concepts of artificial intelligence and reviews existing methods for detecting parking occupancy and traffic congestion using image classification, concluding with approaches to traffic congestion forecasting.
- **Chapter 3** describes two architectural models for classifying parking lot images. The first is based on a customized CNN, while the second applies transfer learning with data augmentation techniques.

- **Chapter 4** proposes a novel method for traffic congestion detection using a parametrized vision transformer with image mapping. It details how video data is transformed through image processing methods.
- **Chapter 5** introduces the Multi-Module Road Traffic Prediction system (MRTE). This chapter explains its architecture and the integration of image processing, vehicle speed analysis, and fuzzy logic to achieve real-time traffic condition assessment.
- **Chapter 6** concludes the thesis, summarizes the key contributions, and outlines possible directions for future work.

Chapter 2 : Background and related work

2.1 Introduction

Image classification is a key task in computer vision that includes automatically assigning labels to images based on their visual content. With the increased availability of visual data in smart cities, image classification has become a significant tool for allowing intelligent surveillance and transportation systems. In many fields, machine learning techniques, particularly Convolutional Neural Networks (CNNs) and Vision Transformers (ViTs), have significantly improved our ability to classify images. Image classification algorithms have been effectively used in the context of intelligent transportation systems to identify parking spot occupancy and assess traffic congestion levels. In parking systems, the goal is to classify each parking spot as either free or occupied using images captured from camera feeds. This offers real-time, non-intrusive monitoring without the need for expensive sensor infrastructure. Similarly, traffic congestion categorization involves studying photographs of roadways to assess the level of congestion, often labeled as light, medium, or heavy. This application offers real-time traffic control and planning in metropolitan contexts.

This chapter offers an overview of artificial intelligence and explores new research linked to its applications in parking occupancy monitoring and traffic congestion estimation based on image categorization. By assessing existing work, we seek to underline the merits, constraints, and research needs that drive the proposed technique in this thesis.

2.2. Artificial Intelligence

At the end of the 20th century, Artificial intelligence had become fully mature and revolutionized the 21st. Artificial intelligence, commonly abbreviated as AI, that creates intelligent machines that can emulate human intelligence, enabling them to think and mimic his actions. Every machine capable of displaying the characteristics of the human mind, such as the ability to learn, the ability to solve problems, the ability to discover meaning and the ability to learn from past experiences, is given this term. The mathematician Alan Turing (Turing 1950), came up with this idea in the 1950s [9]. The question of giving machines intelligence is raised in his book, "Computing Machinery and Intelligence". He wondered if machines could act smart like humans “. . . what we want is a machine that can learn from experience” Alan Turing.

He created a test now recognized as the "Turing Test," where a person talks blindly with another person, then with a machine programmed to make sensible replies, without noticeable differences. If he is not able to distinguish between them, then a machine has passed its test and according to Turing, can indeed be considered "intelligent" [10]. DL is considered a subset of ML, and ML is considered a subset of AI which are illustrated in Figure 2.1.

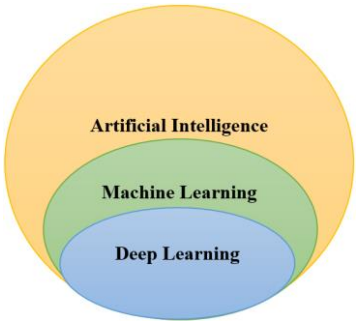


Figure 2.1: Artificial intelligence and its relation with machine and deep learning.

2.3 Machine learning

Machine learning is a subset of AI. The concept of it typically refers to a set of techniques that utilize data to create a model capable of making informed decisions based on what it has learned and, on its experiences [11], reducing the need for human intervention. This definition implies that the model or algorithm needs to go through a learning process in which machines try to learn without explicit programming. This process enables the algorithm to predict new data features that were not part of the initial learning process. This is where supervised methodologies come into play [12,13].

2.3.1 Machine learning Algorithms

Different algorithms are used in machine learning that can be used for solving a diffrent data problems. Logistic regression, KNN, SVM, Decision Tree, Random Forest and Naive Bayes are the most widely used classical algorithms. The selection of algorithms will vary depending on the type of problem to be resolved; for example, regression, classification or clustering.

2.3.1.1 Logistic regression

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. It is preferred for binary targets as it provides predictions ranging from 0.0 to 1.0, representing the probability of the target variable taking on the value of 1. This model is primarily used for classification rather than regression, it is a popular statistical modeling method applied in a variety of industries, including machine learning,

statistics, economics, social sciences, and healthcare, for purposes like identifying spam, analyzing credit risk, diagnosing diseases, and segmenting marketing [14].

2.3.1.2 Decision tree

A decision tree is a graph representing decisions and their results in the form of trees. The nodes in the graph represent an event or a choice, and the edges of the graphs are related to decision rules or conditions. Each tree is composed of nodes and branches. Each node represents an attribute that is to be classified in a group, and each branch's value can be taken by the node [15]. It's called that because it begins as a single or root decision, and then splits into several branches before making any decisions or predictions like trees.

2.3.1.3 Random Forest

Random forests are an ensemble learning method used in classification and regression. It applies the bagging approach to create a number of decision trees with random subset of data. In order to make a final decision tree, the outputs of all decisions trees in any random forest are combined [16]. This classifier uses tow method to make decision: majorly voting or average method.

2.3.1.4 Support vector machines

Another highly popular and advanced machine learning technique used is Support Vector Machine (SVM) which is a supervised learning algorithm that's used for classification as well as regression problems. It's working on a margin calculation concept. In this algorithm, every item of data is plotted as a point in n-dimensional space, where n is the number of features in our database. The corresponding coordinate value is the value of each feature. It classifies the data into different classes by finding a line (hyper plane) which separates the training datasets into classes. It's based on maximizing the distance between the closest data point in each class and the hyper plane that we can call a margin [16].

2.3.1.5 K-Nearest Neighbor

The k-Nearest Neighbour algorithm is commonly used to solve classification problems, although it can apply to regression problems as well. This algorithm leverages a nearest neighbor approach for prediction. It employs a distance metric, such as Euclidean distance and a k value, to calculate the similarity between a new data point and the points within a training dataset. The algorithm then identifies the k nearest ones to the new data point based on the chosen distance metric. Finally, it aggregates the information from these k nearest neighbors, often by calculating the average, to generate a prediction for the new data point [17].

2.4 Deep learning

Deep learning (DL), an evolution of machine learning (ML), which is inspired by the functionality of our brain cells that's based on Deep Neural Networks (DNNs), a form of artificial neural networks (ANNs) with multiple layers of neurons and more complex connections to learn complex patterns and also to provide automatic feature extraction of the input raw data, which minimizes human intervention, unlike traditional machine learning where the features are manually extracted and there is a need to increase the number of neurons if data are too big. Therefore, in order to obtain more accurate results, DL algorithms require a large amount of labeled data for training and also require more time to train due to the complexity of the computational tasks [11]. Figure 2.2 shows how a machine learning system classifies an image by extracting features from the input and using a neural network to decide whether it represents a cat or not.

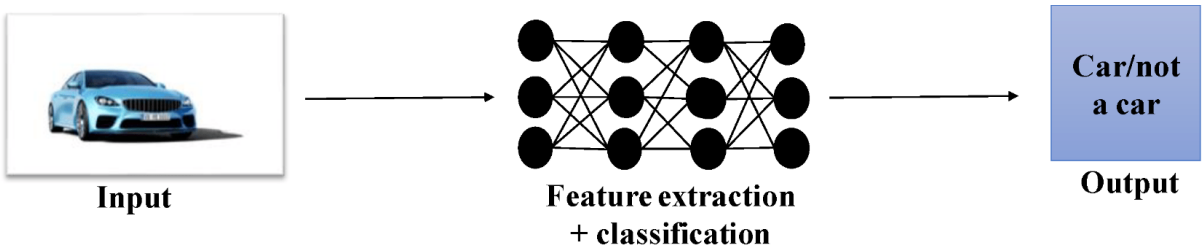


Figure 2.2: Basic Image Classification Pipeline Using Deep Learning.

2.4.1 Traditional Neural Networks

Artificial neurons and perceptrons were inspired by the biological neural network processes. A biological neural network (similar to the one we have in our brain) consists of a large number of nerve cells known as neurons. These neurons connect to one another using axons and dendrites. The Figure 2.3 illustrates these neural connections.

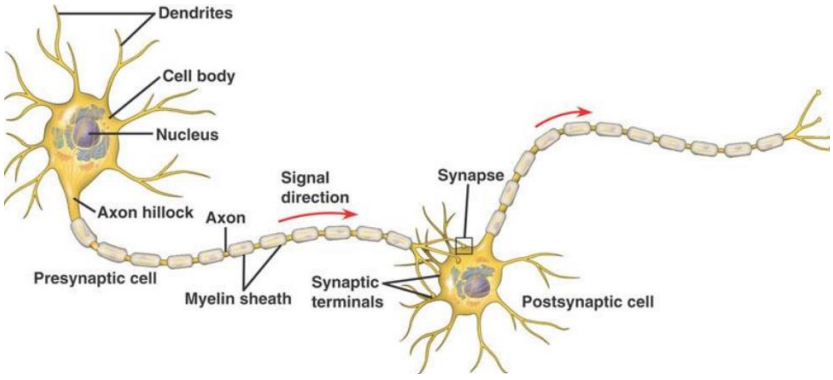


Figure 2.3 : A biological neural network [18].

Artificial neural networks, also called Perceptrons, are reproducing this biological (natural) component, these perceptrons contains computational units referred to as neurons which are interconnected through weights, mimicking the synaptic connections found in biological (natural) organisms. Each input received by a neuron is assigned a weight, shaping the operation performed by that particular unit. Each input to the neuron is scaled with a weight, which has an influence on the function calculated in that unit. The mathematical transformation relationship between the input signal and the output value given by equation 2.1:

$$f = \left(b + \sum_{i=1}^{n} (x_i + w_i) \right) \tag{2.1}$$

f is the activation function, there are many activation functions, such as ReLU, Sigmoid, Tanh, etc, see Section 2.4.3.1.1. X_i is the input signal, n is the number of signals, the weight value of the input signal is W_i, the bias value is $_b$ and the neuron output is Y.

2.4.2 Multilayer Perceptron

Set of neurons, which are organized as networks, have been taken into consideration for the purposes of representing more general regions. MLPs are classical neural networks with one or multiple hidden layers. Typically, the structure begins with an input layer, moves on to multiple hidden layers, and finishes with an output layer. Neurons in one layer can be fully connected to the next layer, known as dense connections. The process of training involves back propagation, which is comparable to how a single neuron works. Specifically, the network receives training data and optimizes weights to minimize the error function between the output and input layers.

In this case, the neurons are organized into three different types of layers:

- **Input layer:** the first layer of the network responsible for receiving data input;
- **Hidden layer:** receives information from a previous and transfers it to the next layer after sum and activation operations. There can be as many hidden layers in a single network as needed;
- **Output layer:** provides the classification answer.

The type and number of layers relate to the architecture of the model. Usually at least two hidden layers are recommended as it has been shown to reduce learning time and improve accuracy [19]. Figure 2.4 shows an example of the structure of MLP.

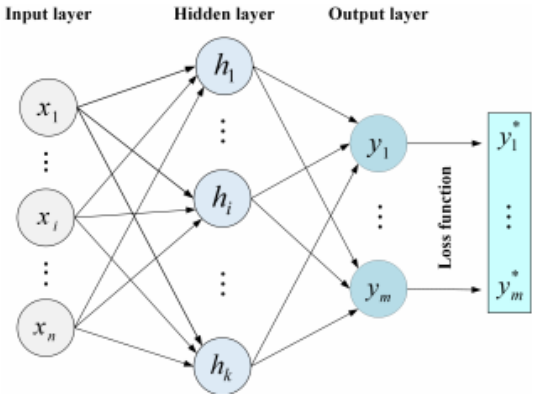


Figure 2.4 : The structure of MLP [20].

2.4.3 Convolutional Neural Network and its Components

The Convolutional Neural Network (CNN) (Lecun, Bottou, Bengio, Haffner, 1998) was initially utilized for the purpose of handwritten character recognition. A CNN represents a specific instance of a deep neural network. Its applications are predominantly in the field of computer vision. The CNN is a multi-layered neural network with a distinctive architectural configuration. This type of network is designed to extract increasingly complex features from the data at each layer, with the objective of determining the output. CNNs have been utilized as low-level feature extractors in an automated manner, circumventing the necessity for manual feature extraction. The convolutional layers within the CNN network apply sliding filters across the height and width of the image. In order to extract specific features such as corners, edges, and textural characteristics, the CNN network is equipped with convolution layers. The scalar product of pixel values and these filters produces a vector containing values that are used in several iterations [20]. CNN architecture: A conventional convolutional neural network (CNN) architecture comprises four principal layers. Figure 2.5 demonstrates a typical architecture of CNN.

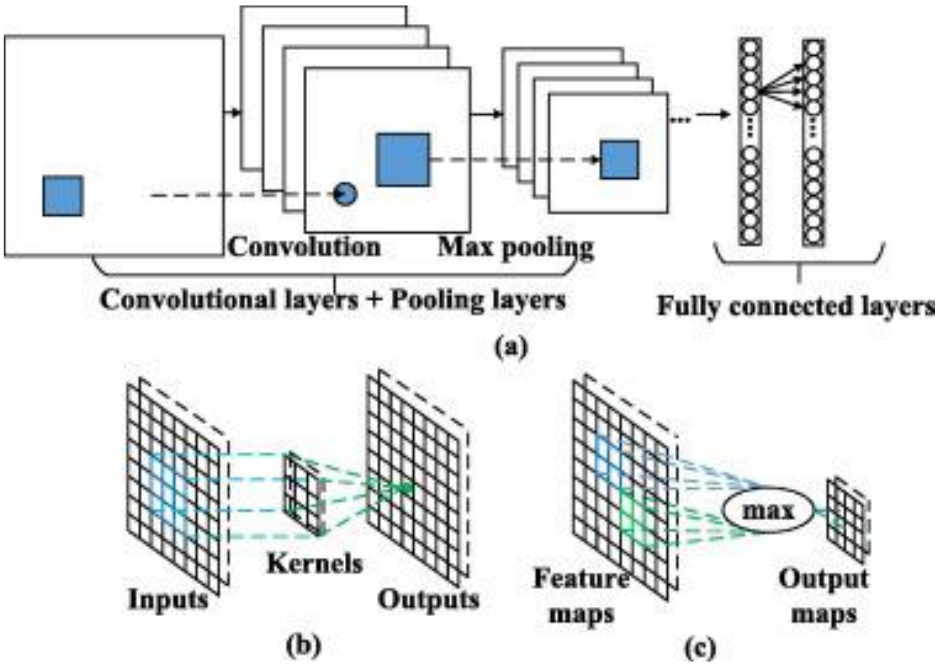


Figure 2.5 : convolutional neural network (CNN) architecture [21].

2.4.3.1 Convolutional layer

The convolutional layer is the initial layer situated above the input image. It represents the fundamental component of the convolutional neural network (CNN) architecture. It is responsible for the convolution operation, which is performed using a set of learnable kernels or filters. These kernels are employed to detect edges, corners, and textures within the input data, thereby extracting features. Feature extraction may involve strides and paddings in conjunction with kernels [22]. A convolution process is presented by figure 2.6.

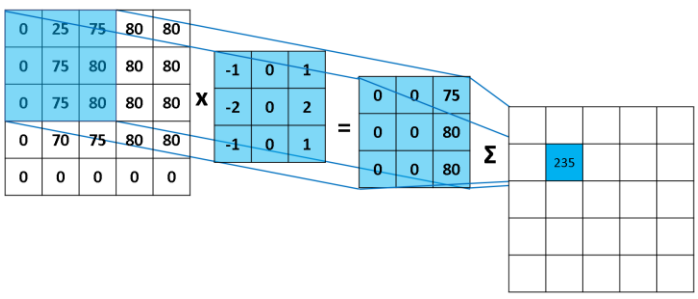


Figure 2.6 : A step in the Convolution Process.

2.4.3.1.1 Activation Function

Activation layers are employed in convolutional neural networks (CNNs) to introduce nonlinearity. They establish the appropriate nonlinear relationship between the input and output signals. This property of non-linearity is introduced through the use of a variety of mathematical functions, which enables the neural network to approximate more accurately. Sigmoid, ReLU and Tanh are three of the most commonly used activation functions. The selection the right activation function is crucial based on the problem at hand as it can impacts how the input data is changed [23].

Sigmoid or logistic: The sigmoid function is the most widely used activation function due to its non-linear nature. This function transforms values from 0 to 1, and it is continuously differentiable. Additionally, the sigmoid function is not symmetric about zero, which ensures that the signs of all neuron output values will be the same [19]. The sigmoid function is defined by formula 2.2.

$$\sigma(x) = \frac{1}{1 + e^{x}} \tag{2.2}$$

Rectified Linear Unit (ReLU): The abbreviation ReLU stands for rectified linear unit, which is a non-linear activation function. One advantage of using the ReLU function is that not all neurons are activated at the same time. This means that a neuron will be deactivated only when the linear transformation output is zero. Another advantage of ReLU is that it is more efficient than other functions because neurons are not activated all at once, but rather a subset of neurons are activated at a time [19]. This function is defined as follows:

$$g(x) = \max(0, x) \tag{2.3}$$

Tangent hyperbolic (Tanh): Tanh function is similar to the sigmoid function This symmetry leads to outputs from previous layers having different signs, which are then passed as inputs to the subsequent layer. The output values of the Tanh function range from -1 to 1 [19]. The formula of tanh function is defined as:

$$f(x) = \frac{e^{x} - e^{x}}{e^{x} + e^{x}} \tag{2.4}$$

2.4.3.2 Pooling layer

The pooling layer is like a data shrinker in a neural network. It takes groups of outputs and replaces them with a single summary value, making the data smaller and easier to manage. This means less computing power and memory are needed. There are different ways to summarize the data, like using the average or the highest value, but the most common method

is max pooling. It simply keeps the largest value from each group, effectively highlighting the most important features [24]. Using the pooling layer can accelerate the computation process and avoid the problem of over-fitting [25].

2.4.3.3 Fully connected layer

The fully-connected layers, also referred to as dense layers, are a component of artificial neural networks (ANNs). These layers play a role in the classification process within CNNs, where they take the flattened output (a vector) from the final convolutional or pooling layer as input. They consist of multiple hidden layers that extract more complex high-level features from the

preceding network. Ultimately, an activation function is applied in the final layer to classify the input into the category with the highest probability. The number of neurons at the end output corresponds to the number of categories, and the resulting output vector is used to determine the category to which the image belongs [19].

2.4.3.4 Loss/classification layer

The loss layer defines how the network's training penalizes the difference between the predicted and actual signals, typically serving as the network's final layer. Different loss functions suited to various tasks may be applied within this layer. Loss Functions refer to mathematical functions that measure the difference between the model's predicted output and the actual output. These functions enable the quantification of the error or cost of the model's predictions, which is then utilized to fine-tune the model's parameters during training in order to minimize this error. In another word, to reduce the validation loss that reflects the model's performance on unseen instances. Therefore, loss functions play a crucial role in directing the learning process and improve the accuracy of the model's predictions. The general form of a loss function is defined as:

$$L(y, \hat{y}) = \frac{1}{N}\sum_{i=1}^{n} l(y_i, \hat{y}_i) \tag{2.5}$$

where: $L(y, \hat{y})$ the overall loss, N is the total number of samples in the dataset. i refer to each sample in the dataset. y_i is the true value. $\hat{y}_i$ is the predicted value and $l(y_i, \hat{y}_i)$ is the loss/error function for a single sample, which measures the difference between the true value y_i and the predicted value $\hat{y}$.

2.4.4 Autoencoders

Autoencoders are a type of deep learning model that creates a coded representation of input data. The initial part of the model generates the code (encoder), while the second part is responsible for reconstructing the original data (decoder). The training quality is evaluated by

comparing the input layer with the output layer. Upon completion of training, the decoder and the output layer are removed, and the values generated by the encoder become the output of the network. Autoencoders have numerous applications, such as reducing dimensionality, retrieving information, and performing various computer vision tasks. In computer vision applications, the encoder and decoder layers are commonly convolutional [19].

2.4.5 Generative adversarial networks (GANs)

A type of neural network architecture known as a Generative Adversarial Network (GAN) was created by Ian Goodfellow, for the purpose of generating models to produce new realistic samples whenever required. This process includes the automatic detection and understanding of patterns or consistencies in the input data in order to create new examples from the initial dataset using the model. GANs consist of a generator *G* that generates new data resembling the original data, and a discriminator *D* that predicts the probability of a new sample coming from real data rather than data generated by the generator. Therefore, both the generator and the discriminator are trained to compete with each other in GAN modelling. While the generator tries to fool and confuse the discriminator by creating more realistic data, the discriminator works to distinguish real data from a fake one generated by *G* [26].

2.4.6 YOLO (You Only Look Once)

Real-time object detection and recognition are crucial for various applications such as traffic management systems, surveillance systems, and self-driving cars [27]. Object detection is the most challenging part of computer vision, since it combines image classification, which identifies the object in the image, and image localization, which determines the object's location in the image [28]. YOLO is a clever convolutional neural network (CNN) for object detection which is the most commonly used real-time object detector at the moment because of its ability to produce more accurate detection results while also being able to run in real-time or used for real-time applications [29,30]. The YOLO series of algorithms has been experiencing rapid growth and is considered one of the top-performing algorithms. Especially, the newly introduced YOLOv8 algorithm in 2023 has notably achieved the highest levels of accuracy so far [31]. Figure 2.7 shows the YOLO models evolution

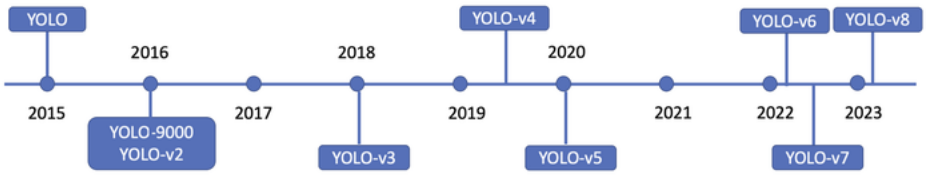


Figure 2.7 : The YOLO models series evolution timeline [32].

2.4.5 Transformers

The transformers were brought out by Vaswani and his team in their seminal 2017 paper "Attention Is All You Need" [33] for machine translation. The paper has changed the areas of natural language processing (NLP) and computer vision, helping to create new state of the art models for solving a wide range of problems. Without the use of recurrences and convolutions, transformers are solely dependent on the attention mechanism. More specifically, the attention mechanism computes a relation mask between words in sentence that is used to determine which words are associated with each other within an encoder decoder structure.

Both an encoder and a decoder are used in natural language processing (NLP): the encoder processes the input sequence (a sentence, for example), and the decoder uses attention to link the two to produce an output sequence (a translation, for example).Vision Transformer (ViT), on the other hand, just makes use of the encoder. To generate a classification output, images are separated into patches, handled like tokens, and then run through the encoder. Image tasks concentrate on feature extraction, negating the need for the decoder, whereas NLP tasks require it for sequence generation. Their main advantage is that they allow models to capture long term dependencies irrespective of the position and distance of the input or output sequences. The architecture of the transformers, consisting of an encoder and decoder set up in a stack of the same layers.

2.4.5.1 Vision Transformers (ViT)

Recent advancements in deep learning have demonstrated remarkable achievements in various computer vision tasks, such as object detection and image classification. A revolutionary advancement is the Vision Transformer (ViT) architecture, which has displayed outstanding results in tasks involving the classification of large-scale images [34].

The way in which an image is represented and processed is a major difference between ViT and other deep learning models for image classification. ViT differs from other image classification models, such as convolutional neural networks (CNNs), in that it represents an image as a set of tokens, rather than using convolutional operations on the whole image. This enables ViT to process large images effectively and to scale up to high resolution images, while maintaining the ability to capture detailed details. The architecture of CNNs uses local patterns no matter where they are located in the image. Vision Transformers (ViT) utilize the attention mechanism to focus on various areas of the image at once, but they may not effectively utilize the image's local features. According to figure 2.8, the ViT image classification process can be divided into the following steps [35]:

- Divide each image into a sequence of flattened 2D patches without overlapping.
- The flattened patches are embedded for linear projection.
- The classification token and a matrix for position embeddings are included
- in the projections of the patches.
- Feed the patch and position embedding matrix into transformer layer.
- The final output prediction is obtained by feeding the transformer encoder output into an MLP head.

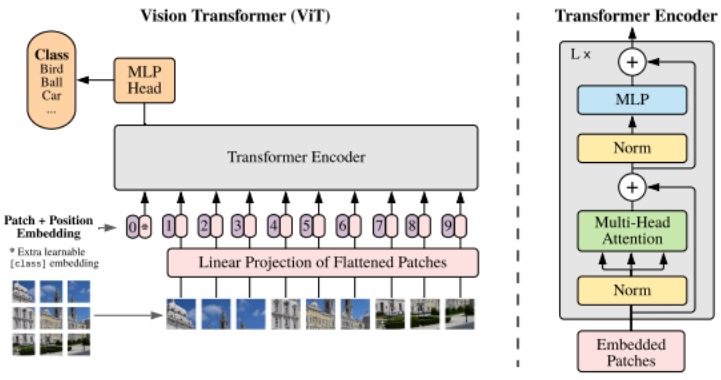


Figure 2.8 : Architecture of the vision transformer from [35].

2.5 SIFT method overview

Scale Invariant Feature Transform (SIFT) [36] is a popular method used to detect and describe local features in images. Due to its robustness and efficiency, SIFT has become a cornerstone in various computer vision applications including object recognition, image fusion, and 3D reconstruction. SIFT includes the following basic features.

2.5.1 Scale space extremum detection

SIFT detects scale and rotation invariant key points by constructing a scale space representation of the image via Gaussian blur at varying scales. This process requires convolving the input image I(x ,y) with Gaussian kernels of different standard deviations (σ) to obtain a scale space representation denoted as L(x, y, σ).

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.6}$$

Here, *(σ)* denotes the scale factor and *G(x, y, σ)* denotes the Gaussian function.

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2}} \tag{2.7}$$

Gaussian differences, calculated as the differences between adjacent scales, help distinguish key points that do not change with respect to scale changes.

$$D(x, y, \sigma) = [G(x, y, z\sigma) - G(x, y, \sigma)] * I(x, y) = L(x, y, z\sigma) - L(x, y, \sigma) \tag{2.8}$$

The parameter z is a constant value that depends on the granularity of the scale space discretization.

2.5.2 Key point localization

After SIFT identifies possible key points, it refines their locations to subpixel accuracy and eliminates low-contrast key points and key points on edges, which are less reliable for matching.

2.5.3 Orientation assignment

SIFT assigns an orientation to each key point according to the local image tilt directions. This step ensures that key point descriptors are invariant to image rotation.

2.5.4 Descriptor generation

SIFT generates a descriptor for each key point to characterize its local appearance. The descriptor is a vector containing information about the slope magnitude and orientation around the key point. These descriptors are robust to changes in illumination, viewing angle, and partial occlusion, making them suitable for matching key points in different images.

2.6 Related Work

In the framework of smart cities, Intelligent Transportation Systems (ITS) have become essential for enhancing urban mobility, reducing environmental impact, and improving the quality of life. Two critical challenges in this domain are the efficient management of parking spaces and the accurate monitoring of traffic congestion. With the growing availability of urban surveillance data, computer vision and deep learning techniques have emerged as powerful tools to automate and optimize these tasks. This section reviews the techniques based on network of sensor nodes and the recent advancements in image classification approaches for both parking lot detection and traffic congestion analysis, highlighting their contributions to intelligent urban transportation solutions.

2.6.1 Image classification approaches for parking lot

2.6.1.1 Methods Based on Network of Sensor Nodes

Many systems are implemented using a network of sensor nodes, typically buried under the asphalt so that each node detects a parking spot. Different types of sensors have already been used. In [37,38], the authors use light sensors and evaluate the performance of their system using remote-controlled toy cars. In [39], the authors utilize a wireless sensor network (WSN) composed of optical sensors, while in [40] a combination of magnetic and ultrasonic sensors was used. In [41], a smart parking monitoring system based on WSN is proposed, and magnetic sensors have been used to monitor the occupancy of a vehicle parking lot in [42]. Also, a particular type of sensor was used for the detection of parking spaces for Berth vehicles in Carpack [43]. In [44], a comparison of the performance of different sensors for intelligent parking is presented.

2.6.1.2 Feature-based approaches

Parking space classification system works, beginning with an overhead image of a parking lot. To create numerical representations (feature vectors called image descriptor) of each parking space, feature extraction is the next step after image pre-processing, which isolates individual parking spaces. A machine learning classifier like a Support Vector Machine (SVM) or Multilayer Perceptron (MLP) is then trained using these vectors and their ground-truth labels (such as occupied or vacant). A trained model that can correctly identify parking spots in fresh, unseen image is the end result. Figure 2.9 shows parking space classification system works based on feature extraction. During the image preprocessing step, some authors may also use techniques like image scaling and histogram equalization to make the image better for feature extraction. During the feature extraction process, you can get one or more feature

vectors from the images. For example, the Local Phase Quantization (LPQ)[45], the Local Binary Patterns (LBP) [46], or the Histogram of Oriented Gradients (HOG)[47].

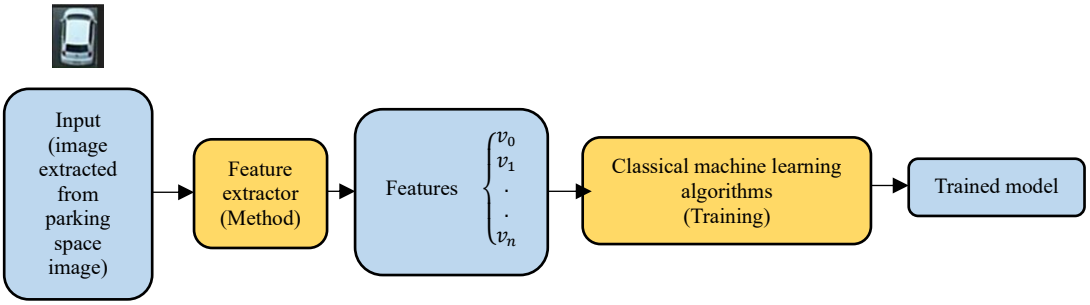


Figure 2.9 : Parking space classification system works based on feature extraction. v_i refers to the feature vector (image descriptor) extracted from parking space image.

When classifying parking spaces, texture-based features such as Quaternionic Local Ranking Binary Pattern (QLRBP), Local Binary Patterns (LBP), and Local Phase Quantization (LPQ) are frequently used [48]. For efficient classification, the authors [49] suggested combining Support Vector Machines (SVMs) with LPQ and LBP descriptors as feature vectors and performed on UFPR4 that a subset of PKLot dataset [41]. By making the entire PKLot dataset available, they built upon their earlier research. For the classification task in this extended study by [50], ensembles of Support Vector Machines (SVMs) trained with LPQ and LBP features were used. Motivated by changes in lighting, camera angles, and environmental conditions, the authors [51,52] used a concept drift approach to classify parking spaces. Using the PKLot dataset in a day-by-day setup, they unveiled the Dynse framework with LBP features in [51] . Models trained on samples from the previous day were used to classify images from the current day. In [53], they expanded the analysis to include more datasets, demonstrating that adaptive techniques perform better than static classifiers when managing concept drift, especially in the PKLot dataset.

The Quaternionic Local Ranking Binary Pattern (QLRBP) is used by [53] as a texture descriptor in color parking space images. They used 6,000 individual images from the UFPR04 subset and used Support Vector Machines (SVMs) and k-nearest neighbors (k-NN) as classifiers. The authors [54,55,56,57] also investigated using LBP-based features using k-NN classifier from the PKLot dataset . In [56] the authors used subsets from the PKLot [49] and CNRPark-EXT datasets [58], including SVM classifiers and examining parking lot variability. Histogram of Oriented Gradients (HOG) and Local Binary Pattern (LBP) is

employed by [59] as feature descriptors for a linear SVM classifier. The authors also used an Adaptive Median Filter (AMF) to apply background subtraction. Their findings demonstrated that the UFPR04 subset of the PKLot dataset performed well when HOG features were used alone. In a related study, [60] used Mean Squared Error (MSE) in conjunction with Uniform Local Binary Pattern (ULBP) on complemented images to classify parking spaces according to a threshold applied to the MSE output. Although the image selection procedure was not made explicit, they tested their approach on 1,000 photos from the PKLot dataset in a range of weather conditions.

Using subsets from the PKLot [49] and CNRPark [61] datasets as well as a new dataset that comprised images from CNRPark and ImageNet [62], [63]created SVM and k-NN classifiers using texture features gathered at several image scales. The authors [64] confined their research to 60 images from the PUCPR subset of PKLot and applied Gray-Level Co-Occurrence Matrices (GLCM) as texture descriptors to categorize parking spaces as filled or empty based on resemblance to training images. Although they did not specify the testing approach. In their research, [65] combine texture and color features for analysis. Through experimentation with the PKLot dataset, they assess various machine learning models, such as Neural Networks, SVMs, k-NNs, and Naïve Bayes classifiers. [65] merged color and texture features and assessed numerous classifiers, such as Neural Networks, SVMs, k-NNs, and Naïve Bayes, using the PKLot dataset. Using pixel values from different color spaces is another popular method that has been investigated by researchers [66,67]. Histograms in the Hue, Saturation, and Value (HSV) color space were computed by [67] directly in smart cameras. After being sent to a central server, these histograms were utilized as input features for a linear SVM classifier, a technique also detailed in a related study by [68]. Their tests, which used the PKLot dataset, showed that energy and bandwidth consumption can be decreased by processing images on-device and transmitting only the feature vectors or compressed images. The performance of SVM and Logistic Regression classifiers trained on feature channels that is, the distinct color channels of an image was also assessed by [66] using the PKLot dataset. By [69], the authors presented a system that uses Raspberry Pi Zero camera modules in conjunction with a computationally effective occupancy detection algorithm that is based on a Support Vector Machine (SVM) classifier and the Histogram of Oriented Gradients (HOG) feature descriptor.The PKLot and CNRPark-EXT datasets were used in experiments to perform classification by applying a threshold to this descriptor. In a different study, [70] classified individual parking spaces using chromatic

gradient analysis and improved performance by introducing an adaptive weather analysis method. The PUCPR subset of the PKLot dataset was used for classification, which was also based on a threshold. Using the Canny edge detector [71], [72] improved their method by converting images into the LUV color space for feature extraction, which was followed by Random Forest algorithm classification.

Bag-of-features representations were used by [73] and [74] to categorize parking spaces. [65] used an SVM classifier to classify parking spaces in the PKLot and CNRPark-EXT datasets by combining color features with Speeded Up Robust Features (SURF)[75]. Similarly, [74] used a bag-of-features model with an SVM with a radial basis function kernel for classification and SIFT (Scale-Invariant Feature Transform) [36] for feature extraction. Using the PKLot dataset, their evaluation took into account changes in camera angle, weather, and parking lot environments. They also suggested a deep learning-based approach, which is covered in more detail in the next section.

2.6.1.3 Deep learning approaches

Although they integrate feature extraction and classifier training into a single step of the deep learning algorithm, deep learning (DL) techniques function similarly to feature-based techniques. Since deep learning models are designed to automatically learn how to display parking spaces, this approach eliminates the need for manual feature engineering. Additionally, the classifier is typically integrated into the DL model. Figure 2.10 shows the operation of this technique.

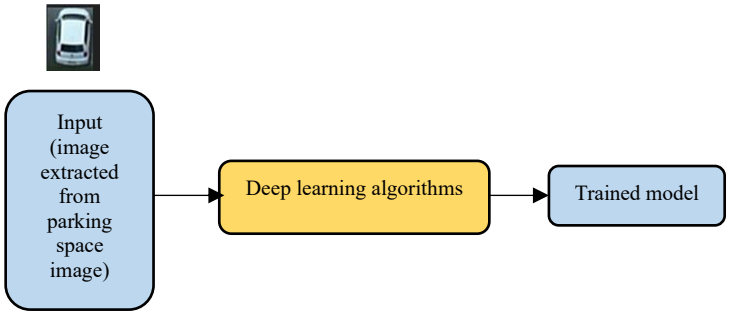


Figure 2.10 : Overview of deep learning-based approaches.

Recent years have seen the development and enhancement of numerous computer applications and systems in a variety of research fields, such as computer vision, pattern recognition, and big data analytics, thanks in large part to machine learning (ML) and deep learning (DL) techniques [76.77] . Convolutional Neural Networks (CNNs) are among the best deep

learning techniques for vision-based detection and segmentation tasks because of their demonstrated high efficacy in these tasks [78,79,80,81]. AlexNet a deep convolutional neural network (CNN) architecture presented by [82]. It was created to classify images and has eight layers, three of which are fully connected and five of which are convolutional. Many searchers used AlexNet in the context of parking image classification, the study [83] examined supervised and semi-supervised learning techniques. They employed a semi-supervised technique known as pseudo-label for the other method and AlexNet, for the supervised approach. they tested both models using Pklot dataset. The experiment's findings demonstrated that AlexNet consistently outperformed the pseudo-label approach across all evaluation metrics. A modified version of the AlexNet architecture is used by [61] and performed on CNRPark dataset, An expanded version of the dataset, CNRPark-EXT was used by [58] , in a follow-up study. Tests conducted on the PKLot dataset showed that their mAlexNet model was resilient to changes in parking lot conditions and camera angles, exhibiting only slight accuracy declines in a variety of scenarios. Later, using the CNRPark-EXT dataset for evaluation, the authors [84] used mAlexNet to classify parking spaces using smart cameras. mAlexNet is used by [85], but they did not noticeably improve performance by changing the first convolutional layer's kernel size. A Camera A from the CNRPark dataset as well as a private dataset is used by [86] to compare mAlexNet, AlexNet, and MobileNet [87]. They did not specify the cost or duration of post-capture image processing, despite the fact that their strategy focused on inexpensive hardware. A lightweight version of the AlexNet architecture was also presented by [88] , who experimented with a private dataset in addition to subsets of the PKLot and CNRPark-EXT datasets. Nevertheless, information about the private dataset's accessibility and the subsets' selection standards is lacking. LeNet [89] and AlexNet were used [90], which are well known for their small architectures, to classify parking lots. The PKLot dataset and a private dataset are both used in the study; however, the testing process used on the PKLot data is not explained in detail.

The authors [91] et al propose a low-complex deep neural network architecture using a lightweight model for a smart parking occupancy detection system. They use a Raspberry Pi 3 camera to detect open parking spots, and the performance of the model is assessed using the PKLot dataset after training and cross-validation. In [92], they define a system as two modules working together. The first module employs the YOLO V3 [93] architecture to detect the license plate, while for parking detection and parking spot selection, the second module uses the mAlexNet architecture and used by PKLot and CNRPark-EX.

A popular technique is transfer learning [94], where a network is first trained on a general-purpose dataset and then refined using a PKLot dataset. In this context [95] refined a VGG16 model, that they had trained on the ImageNet dataset [62]. In a similar vein, [96] presented a customized network and a modified VGG16 architecture [97] for parking space occupancy classification. Additionally, they used the PKLot dataset for testing and applied image transformations to produce top-down views of the parking lots. Another study is provided by [98], where the authors use a camera system and the original VGG16 network model to estimate parking lot occupancy in real time with a camera system (CCTV). The model was trained and tested using the PKLot and CNRPark-EXT datasets. [99] suggested a technique in which an SVM classifier is fed features that were extracted from a VGG16 model. Five-fold cross-validation on PKLot images was used for their evaluation. A CNN model with a fixed input size of 40 × 40 pixels was presented by [100] and assessed using the PKLot dataset's original testing protocol. A CNN-based method using dilated convolutions which skip specific pixels within the convolutional kernel was presented by [101]. The authors claim that this design produced encouraging results and improved the model's capacity to represent the images' global context.

In order to facilitate the extraction of more intricate features, the authors [102] suggested augmenting the YOLOv3 architecture [103] by adding residual blocks. Images of parking lots were classified using the modified network. It was first fine-tuned using a subset of images from the PUCPR section of the PKLot dataset after being trained on vehicle images from the PASCAL VOC [104] and COCO [105] datasets. To classify individual parking spaces, a reduced-depth version of the ResNet50 architecture [106] used by [107] The PKLot dataset and a private dataset were used in their experiments, and stratified sampling was used to separate the training and testing samples. The original ResNet50 network was used by [108] in a related study, evaluating it on small subsets of the PKLot and CNRPark-EXT datasets.

The model proposed by [109] presents a deep learning-based framework. They tested two main approaches: firstly, using ResNet as a feature extractor and a binary SVM classifier to classify parking spaces as occupied or vacant. Secondly, they combined VGG16 with OpenCV functionalities. They also evaluated their approach on the PKLot dataset. The results demonstrates that ResNet with SVM achieved the highest accuracy.

2.6.1.4 Car park datasets

This section goes into great detail about the publicly available parking lot datasets we used in our related study, such as PKLot, CNRPark, and CNRPark-EXT.

2.6.1.4.1 PKLot dataset

When [49] first released the PKLot dataset, it contained pictures from a single parking lot taken from a single camera angle. The more recent and thorough version, which was published by [50] , includes 12,417 photos captured from two separate parking lots from three different camera angles. In particular, the Federal University of Paraná (UFPR) building's fourth and fifth floors were used to record the UFPR04 and UFPR05 subsets. Despite showing the same parking lot, the camera angle and the dates of capture differ between the two subsets. Images from the tenth floor of a building at the Pontifical Catholic University of Paraná, which represents a different parking lot and viewing perspective, were used to create a third subset, PUCPR. The dataset contains 695,851 manually annotated parking space images in total, of which 51.4% (358,071) are marked as vacant and 48.6% (337,780) as occupied. During the day, images were taken every five minutes, and each frame was labeled with the weather type (cloudy, rainy, or sunny). PKLot dataset samples are shown in Figure 2.11.

Figure 2.11 : PKLot dataset samples.

2.6.1.4.2 CNRPark and CNRPark-EXT datasets

Images taken from a single parking lot from two distinct camera angles make up the CNRPark dataset, which was first presented by [61]. Two distinct sessions were used for each camera angle, and the data was gathered at 5-minute intervals during the day in sunny conditions. Of the 12,584 annotated parking spots in the dataset, 4181 (33.2%) are marked as free, and 8403 (66.8%) are marked as occupied. Public access is limited to cropped (segmented) parking space images, all of which were resized to 150 × 150 pixels.

Later, [58] released an improved version called CNRPark-EXT. Nine unsynchronized cameras that took pictures every 30 minutes while observing the same parking lot from

different perspectives were added to the dataset. There are 4278 full-scene images in JPEG format, each measuring 1000 × 750 pixels. Additionally, there are 150 × 150 pixel segmented parking space images available. Figure 2.12 shows CNRPark+EXT dataset samples.

Figure 2.12 : CNRPark+EXT dataset samples.

2.6.1.5 Summary and comparison of classification approaches

A summary of feature extraction-based parking lot classification techniques is shown in Table 2.1, Table 2.2 provides a summary of techniques based on deep learning (DL). The main metric used to assess parking space classification is accuracy. Even though every method in the Overview makes use of at least one public dataset. Since the majority of authors used this metric to present their findings, we also report accuracy.

Table 2.1 : Classification of parking spaces using feature extraction methods.

Authors	Year	Classifier and features	Accuracy (%)	Datasets used
[49]	2013	Ensembles of SVM and LBP/LPQ	99.8	PKLot (UFPR04)
[50]	2015	Ensembles of SVM and LBP/LPQ	99.6	PKLot
[67]	2015	SVM and HSV Histogram	96.0	PKLot
[66]	2016	SVM/LR and Feature Channels	99.9	PKLot
[59]	2017	Background subtraction with AMF and linear SVM/LBP with HOG	21,0	PKLot (UFPR04)
[53]	2018	k-NN/SVM and QLRBP	99.4	PKLot
[51]	2018	Dynse Framework and LBP	92,2	PKLot
[69]	2018	SVM and HOG	93.2	PKLot
[54]	2018	k-NN and LBP-based features	93.2	PKLot
[55]	2018	k-NN and LBP-based features	98.6	PKLot
[74]	2018	SVM and SIFT (Bag of Features)	92.0	PKLot
[63]	2019	SVM/k-NN and textural features	99.6	PKLot, CNRPark-EXT
[72]	2019	Canny and color features with Random Forest	98.31	PKLot
[56]	2018	k-NN/SVM and LBP-based features	85,6	PKLot, CNRPark-EXT
[60]	2019	Threshold and ULBP	79	PKLot
[73]	2020	SVM and SURF	99.7	PKLot, CNRPark-EXT
[64]	2020	Euclidean Similarity and GLCM	95	PKLot(PUCPR)
[57]	2020	k-NN and LBP-based features	87	PKLot(UFPR04)
[65]	2020	Neural Networks, k-NN, SVM, Naïve Bayes, textures and color features	99.7	PKLot
[52]	2020	Approaches for concept drift and LBP features	90.4	PKLot

Classifiers, feature extraction techniques, attained accuracy, and datasets used are highlighted in the Table 2.1, which offers a comparative overview of research studies on feature-based approaches for parking lot occupancy detection. The use of Local Binary Patterns (LBP) and its variations (ULBP, QLRBP, etc.) in conjunction with classifiers like Support Vector Machines (SVM) and k-Nearest Neighbors (k-NN), which routinely attain high accuracy (e.g., 99.8% in [49], 99.9% in [66]), is a prevalent trend in the literature. These techniques make use of texture-based descriptors since they are reliable at identifying parking space occupancy trends. Approaches that rely on thresholding techniques [60] or background subtraction with AMF [59] perform significantly worse (21% and 79%, respectively), which is indicative of their limitations when dealing with complex lighting conditions and dynamic environments.

Although some studies extend evaluations to CNRPark-EXT and PUCPR subsets for cross-validation, the PKLot dataset is still the most commonly used benchmark. While more recent works like [65] integrate neural networks with traditional classifiers, achieving accuracies as high as 99.7%, hybrid methods that combine multiple features (e.g., LBP with HOG [59], color histograms [67], or SIFT [74]) show improved versatility. However, there are still issues with concept drift [52] and scalability in practical applications, as demonstrated by the poorer performance of models that only use edge detection (e.g., Canny [71]) or single-feature descriptors.

All things considered, the table highlights how well ensemble methods and multi-feature fusion work to achieve state-of-the-art outcomes while also highlighting weaknesses in resilience and environmental variability adaptability. These drawbacks support the shift to deep learning techniques, which eliminate the need for manually created descriptors by automatically learning discriminative and transferable feature representations straight from raw images. Building scalable and intelligent parking management systems within smart city frameworks is a better fit for deep learning models, especially convolutional neural networks (CNNs), which provide greater adaptability to a variety of real-world conditions and large-scale data.

Table 2.2 : Classification of parking spaces using deep learning methods

Authors	Year	Deep learning model	Accuracy (%)	Datasets used
[61]	2016	mAlexNet	99.6	PKLot, CNRPark-EXT
[95]	2016	VGG16	100	PKLot
[83]	2016	AlexNet	99.8	PKLot
[58]	2017	mAlexNet	98.1	PKLot, CNRPark-EXT
[100]	2017	Designed CNN	99.9	PKLot
[90]	2017	AlexNet and LeNet	99	PKLot
[99]	2018	VGG16 / SVM	99.7	PKLot
[88]	2018	Custom AlexNet	99.5	PKLot, CNRPark-EXT
[84]	2018	mAlexNet using Background modeling and Canny edge detector	99.6	CNRPark-EXT
[101]	2019	CarNet	98.8	PKLot, CNRPark-EXT
[91]	2019	mAlexNet	88%	PKLot
[107]	2019	Custom ResNet	99.9	PKLot
[96]	2019	Custom VGG16 and Custom CNN	99.9	PKLot
[102]	2019	Yolov3	93.3	PKLot
[108]	2020	ResNet50	99.8	PKLot, CNRPark-EXT
[85]	2020	CmAlexNet	99.12	CNRPark
[98]	2021	VGG16	95.7	PKLot, CNRPark
[86]	2021	mAlexNet, AlexNet and MobileNet	97.3	CNRPark
[92]	2021	mAlexNet	90.1	PKLot, CNRPark-EXT
[109]	2024	ResNet / SVM, VGG16	99.7	PKLot

With the majority of studies reporting values above 95% and several surpassing 99%, the results compiled in Table 2.2 show that deep learning models have continuously achieved very high accuracy rates in parking occupancy detection. Strong baselines were set by early contributions (2016–2017) that relied on AlexNet [83], mAlexNet [61], and VGG16 [95], as well as CNN architectures [100]. Accuracy levels, especially on the PKLot dataset, approached or exceeded 100%. Custom CNNs [88], VGG16/SVM hybrids [99], and deeper architectures like ResNet [107] and custom VGG16 models [96] were introduced in later works between 2018 and 2019. These not only maintained but also, in certain cases, enhanced performance, reaching up to 99.9%. Some models, however, like Yolov3 [102] (93.3%) and mAlexNet [91] (88%) reported significantly worse results, indicating how sensitive performance is to network design decisions and dataset complexity. With accuracies near 100% across PKLot and CNRPark(-EXT), ResNet50 [108], CmAlexNet [85], and hybrid models that combine ResNet or VGG16 with SVM classifiers [109] confirmed the dominance of deep architectures starting in 2020. Although accuracy saturation has been mostly attained on standard datasets, the analysis shows that improving robustness and generalization to a variety of real-world scenarios outside of benchmark conditions remains a challenge.

2.6.2 Approaches for traffic congestion detection

2.6.2.1 Traditional vision-based methods for traffic congestion detection

Video cameras have become integral in traffic congestion estimation due to their non-intrusive installation on road infrastructure. Analyzing traffic congestion generally requires two key elements: vehicle density and speed. Various studies have explored methods to estimate these factors effectively. Vehicle density can be estimated through object detection using features such as colors, pixels, and edges, and vehicle identification techniques that include various machine learning methods [110,111]. A traffic control system model for intersections is developed by (2017) [112] using background subtraction and Haar Cascade to collect real-time traffic data, including vehicle density and other statistics. Digital solutions are then used to transmit this data. Background subtraction techniques have been used to identify moving objects and forecast future congestion by analyzing their edges using the Canny algorithm. These techniques were implemented using OpenCV software by [113,114]. CCTV camera analysis and image processing techniques to investigate traffic congestion are used by [115]. Three traffic status outputs flow, dense, and congested are produced by the method, which consists of traffic density estimation, traffic status classification, and vehicle detection using background subtraction.

Other methods such as adaptive background subtraction is used in video analysis to detect traffic congestion by converting videos to grayscale images, binarizing them, and applying noise reduction techniques by [116]. In addition, an algorithm has been proposed to evaluate traffic congestion at urban intersections [117]. This algorithm works in two stages: global vehicle speed detection and traffic situation identification. In the first stage, Shi-Tomasi corner detection method [118] and Lucas-Kanade optical flow algorithm [119] are used to estimate vehicle speed. In the second stage, traffic situations are identified by analyzing the global vehicle speed during a single traffic light cycle. In the context of detecting corner points and estimating vehicle speed, the authors [120] used a simple rule that evaluates the level of traffic congestion by integrating traffic density and speed parameters. Online images have also been used to develop real-time congestion detection systems. According to the image correlation coefficient threshold between consecutive images, one such system by [121] uses Haar Cascade for vehicle detection and classifies congestion into two states: normal and congested. Congestion detection algorithm using texture analysis through image analysis techniques is developed by [122]. The algorithm consists of five steps: video frame acquisition, vehicle area calibration, Gray-Level Co-occurrence Matrix

computation [123], feature extraction, and road congestion recognition. However, in partially occluded objects, texture features may decrease reliability.

Other methods use image texture analysis to estimate congestion on urban roads. In [124], the authors propose a method that involves first smoothing images using plane-to-plane homograph to recognize cars. Then, SIFT (Scale Invariant Feature Transform) [125] feature detection and matching is performed, and then the Random Sample Consensus (RANSAC) algorithm is used to estimate the homograph matrix. The method uses Sum of Absolute Differences (SAD) to detect moving vehicles and uses SIFT feature matching to track these vehicles. The method proposed by [126] automatically categorizes spatial and temporal traffic patterns from network sensor data using contour detection, the feature bag method with Accelerated Robust Features (SURF) [127] descriptors, and the multiclass Support Vector Machine (SVM, an effective method for object detection and has been used in various vehicle detection) classifiers. Haar feature based cascade classifiers, first proposed by [128], are other systems [129]. A study comparing macroscopic and microscopic methods for estimating road traffic density was carried out by [130] . Road traffic was divided into light, medium, and heavy categories using three classifiers: K-Nearest Neighbor (KNN), Learning Vector Quantization (LVQ), and Support Vector Machine (SVM). The objective was to create an extremely precise traffic density estimation algorithm. [131] classify traffic data with Auto-Regressive Stochastic Processes (ARSP) and SVM, achieving reliable classification. The approach's high computational demand, however, limits its feasibility for real time applications.

Using histograms of space-time orientation structure distributions, a dynamic texture classification technique is presented by [132] that does not require motion estimation, tracking, or segmentation. The technique showed good classification performance and resistance to common environmental conditions; however, noise, object speeds, and video sequence frame rate can affect the precision of spatiotemporal orientation analysis, which could make it difficult to produce trustworthy results. Dynamic texture recognition for traffic classification is used by [133]. This method represents object motion with a graph, where nodes represent vehicles, and edges define the motion relationships between them. The findings showed that this method could accurately classify various traffic scenarios, surpassing several existing dynamic texture recognition methods.

A statistical method based on motion vectors to classify highway traffic congestion is offered by [134]. The method employs the pyramidal Kanada-Lucas-Tomasi (KLT) tracker algorithm. After extracting motion data, traffic patterns were categorized into three groups:

light, medium, and heavy. The classification, conducted using neural networks, exhibited resilience to environmental factors such as fluctuating luminance. However, no results were presented concerning occlusions and sensitivity to noise. A method based on the holistic properties of traffic videos is presented in [135]. The technique classifies the level of traffic congestion after first extracting holistic features from the video, such as motion energy, color, and texture. In their study, [136] investigate the feasibility of traffic monitoring in the absence of motion features. An approach is proposed to process ultra-low frame rate films or videos where accurate motion characteristics cannot be derived. The assessment of traffic state, such as fluid, dense, or congested, relied on 2D spatial characteristics and a machine learning algorithm. Some researchers have investigated alternative traffic monitoring approaches that do not rely on vehicle identification or tracking [137]. One of these methods uses a camera-based technique to monitor and predict traffic without the need for individual vehicle detection. It analyzes video streams to extract spatial interest points (SIPs) indicating the number of vehicles and spatial-temporal interest points (STIPs) reflecting the number of moving vehicles. These features are then classified using a Gaussian mixture model, and a dynamic Bayesian network is used to learn road condition patterns and predict future traffic conditions.

The authors of the study [138] initially proposed YOLO and Haar Cascade as the best-performing car detectors. They have been selected to facilitate the extraction of visual features in order to construct the input vector for vehicle classifiers using various machine learning classifiers, including Random Forest, SVM, and KNN. Second, the authors used ResNet [139] and the deep network architecture suggested in [140] and tested them on two classes (Heavy and Light) for traffic state classification before expanding them to three classes. (Light, medium, and heavy). When it comes to accuracy, the deep learning method outperforms the conventional method.

2.6.2.2 Deep learning methods and transformer based for classifying traffic congestion

The problem of tracking highway traffic with slow-frame-rate videos, where motion features are unreliable for analysis, is addressed by [141]. Instead of traditional traffic condition measurements, this method aims to identify them by capitalizing on the repetitive nature of topdown traffic scenes. The homogeneous content of traffic scenes allows for focused analysis on unique elements in such images. Operating without motion characteristics, this method segments traffic photos, categorizes traffic scenes, and calculates traffic density. It proposes distinct Convolutional Neural Network (CNN) models for categorizing traffic photos into different groups (empty, fluid, heavy, jam), segmenting traffic images into three

classes (road, automobile, background), and estimating traffic density. However, this approach may not perform optimally in non-ideal conditions, such as significant variations in lighting or camera angles, which can impact video quality.

A methodology is proposed by [142] for involving categorizing traffic activity in videos through a structured process comprising three key steps: vehicle monitoring, feature extraction, and classification. In the vehicle monitoring phase, an object detector based on a convolutional neural network coupled with a multi-object tracker was used to accurately identify vehicles. Subsequently, in the feature extraction stage, the authors utilized information gathered from each detected vehicle at various positions along the road to characterize the traffic situation based on three crucial features: density, flow, and velocity. As deep neural networks have advanced, many researchers have endeavored to tackle the problem of traffic congestion classification through a variety of deep learning techniques.

Utilizing convolutional neural networks (CNNs) such as AlexNet, VGGNet, and Support Vector Machine (SVM), [143] introduced a novel classifier named TrafficNet. TrafficNet is developed in four variations: AlexNet+SVM, transfer learning from AlexNet, VGGNet+SVM, and transfer learning from VGGNet. These four models are employed for training and testing images depicting both congested and non-congested traffic scenarios. The proposed method achieves an accuracy level of 90% on the image database compiled using the existing surveillance system operated by the Shaanxi Province Traffic Management Bureau. Traffic images are sourced from hundreds of surveillance cameras, captured at 10-minute intervals. While the TrafficNet name model was also used by [139], their approach differed significantly. They employed a pre-trained and fine-tuned network based on residual learning. Specifically, they leveraged the ResNet-34 architecture to build their deep residual TrafficNet model, designed to classify traffic scenes as congested or not congested. Using a range of scenes captured by the Shanxi Province traffic monitoring system, the CNN-based method for traffic congestion detection was introduced by [144]. To optimize performance and reduce the need for extensive manual labeling, a semi-supervised learning approach is employed in conjunction with a range of classification techniques.

A new technique for categorizing traffic videos is proposed by [145] , the authors employ color coding before training a deep convolutional neural network with traffic data. They utilized the You Only Look Once (YOLO v3) algorithm [146] for vehicle recognition following the conversion of video data into image data. The image dataset was subsequently transformed into a binary image dataset using a color-coded scheme. Finally, these binary images were fed into a Deep Convolutional Neural Network. A recent method proposed by

[147], that uses a causal graph-based representation to identify and manage traffic congestion patterns. This approach extracts key features from traffic data and provides customizable retrieval and similarity measurement of traffic patterns by constructing a causal relationship graph. Another method by [148] introduces a vehicle information feature map (VIFM) combined with a multi-branch convolutional neural network (MBCNN) model to detect traffic congestion using camera image data. This approach aims to minimize the need for additional hardware investments by utilizing the existing camera network in transportation systems. Additionally, [149] investigates various deep learning-based computer vision techniques for traffic environment perception, including object detection, image segmentation, and trajectory estimation, and the use of CNNs and transducers to analyze traffic videos and images to improve congestion detection and management.

Multi-head Self-Attention Vision Transformer (MSViT) used by [150] to study a new macroscopic method for classifying road traffic congestion. While Vision Transformers (ViTs) outperform CNNs on image recognition tasks, the training of ViTs is often challenged by small datasets, thus demanding extensive data augmentation, preprocessing, and careful parameter tuning. The above studies have made significant progress in traffic video analysis. However, limitations remain in terms of adaptability to diverse traffic conditions, computational efficiency, and real-time applicability. While some methods excel in accuracy, they often struggle with environmental robustness, occlusion handling, and extensive data requirements.

2.6.2.3 UCSD Dataset

It is daytime highway traffic videos that contains 254 brief covering light, medium, and heavy congestion can be found in the publicly accessible UCSD (University of California San Diego) dataset [151]. All videos are shot by the Washington State Department of Transportation using a stationary camera above the I-5 highway in Seattle over the course of two days (from May 8, 2004 to June 8, 2004) in a wide range of weather conditions, including overcast, rainy, and clear (sunny). Some of the videos also feature drops on the lens. The videos in this dataset are approximately 5 seconds long and show a frame rate that ranges from 42 to 52 frames per second (fps) at a resolution of 320 x 240 pixels. Three representative images representing three different levels of traffic congestion: Heavy, Medium, and Light that were taken from the Trafficdb dataset are shown in Figure 2.13.

Figure 2.13 : Three images from UCSD illustrating traffic conditions categorized as Heavy, Medium and Light.

2.6.2.4 Comparison of traffic classification accuracies using the UCSD dataset

Comparison accuracy of several methods for detecting traffic congestion is shown in Table 2.3, including transformer-based models, deep learning (DL) techniques, and conventional vision-based methods. Since accuracy is the most widely used metric by most authors to present their findings for assessing traffic congestion classification, we also report accuracy even though all of the methods listed use the UCSD public dataset.

Table 2.3 provides a comparative analysis and highlights the current state of research in this field using the UCSD dataset.

Authors	Method	Image preprocessing employed	Using YOLO	Dataset	Accuracy (%)
[131]	Auto-Regressive Stochastic Processes (ARSP)+SVM (support vector machine classifier)	-	No	USCD	94.50
[135]	holistic method	Region of Interest (ROI)	No	USCD	94.50
[132]	Spatiotemporal filtering formulation using dynamic texture classification	-	No	USCD	95.30
[130]	Microscopic and Macroscopic Parameters of the traffic are extracted (Traffic velocity, Road occupancy rate and Traffic flow)+SVM(support vector machine classifier)	-	No	USCD	96.37
[133]	complex network approach based on graph theory and statistical mechanics + SVM (support vector machine classifier)	-	No	USCD	97.24
[141]	Convolutional Neural Network (CNN) + support vector regression (SVR)	-	No	USCD	97.64
[145]	Deep Convolutional Neural Network (DCNN)	Region of Interest (ROI)+Color-coding	YOLO v3	USCD	98.20
[138]	-YOLO and Haar Cascade with Random Forest, SVM, and KNN. - ResNet and the deep network architecture	-	YOLO v3	UCSD	84 98.6
[142]	Faster R-CNN+ Deep Simple Online and Realtime Tracking (DeepSort) + Random Forest (RF)	-	No	USCD	98.82
[150]	Vision Transformer	-	No	USCD	99.76

The table 2.3 presents a methodical comparison of traffic congestion detection techniques, showing the shift from conventional machine learning methods to sophisticated deep learning and transformer-based models. On the USCD dataset, early methods like Auto-Regressive Stochastic Processes (ARSP) in conjunction with SVM (94.5%) [131] and spatiotemporal filtering with dynamic texture classification (95.3%) [132] relied on manually created features and statistical formulations and produced results with a moderate level of accuracy. Performance was further enhanced (96.37%–97.24%) by combining macroscopic traffic parameters (such as velocity and occupancy rate) with SVM [130] and graph-theoretical techniques [133], proving the usefulness of feature engineering in traffic analysis. With CNN-based models like CNN+SVR (97.64%) [141] and Deep Convolutional Neural Networks (DCNN) using YOLOv3 and preprocessing techniques like Region of Interest (ROI) and color-coding (98.2%) [145] achieving higher accuracy, the advent of deep learning represented a significant advancement. This is because these models are capable of learning discriminative spatial and temporal features on their own. By combining object detection with tracking and classification, hybrid frameworks such as Random Forest (98.82%) and Faster R-CNN with DeepSort [142] further improved performance. Interestingly, the Vision Transformer (ViT) attained the highest accuracy (99.76%) [150], demonstrating the increasing effectiveness of self-attention mechanisms in tasks involving traffic congestion. Nonetheless, the uneven results of YOLOv3-based techniques (84%–98.6%) [138] indicate that preprocessing procedures and classifier choice (e.g., SVM vs. Random Forest) have a significant impact on results. Although these findings show impressive advancements, previous research has highlighted concerns regarding generalizability due to the sole dependence on the USCD dataset [130,145]. To ensure practical applicability, future work should focus on real-world deployment challenges, like computational efficiency and adaptability to dynamic traffic conditions, as well as cross-dataset validation.

2.7 Conclusion

Smart cities' Intelligent Transportation Systems (ITS) concept has advanced significantly, moving from traditional methods to sophisticated deep learning and transformer-based models that can identify parking spaces as occupied or empty and determine traffic congestion levels. This is especially true for parking lot management and traffic congestion detection. Methods like customized CNNs, transfer learning and attention mechanisms have significantly improved model optimization and classification accuracy. In the ITS industry, implementing these architectures in real-time situations is still remains a suitable solution. The use of machine learning and deep learning algorithms and for feature extraction has increased the

accuracy of both parking lot detection and traffic congestion analysis. Notwithstanding these developments, there are still issues to be resolved, such as the requirement to improve detection accuracy, strengthen lightweight models, and overcome constraints in current datasets.

Our research fills these gaps by using customized deep learning, transfer learning, computer vision techniques and transformer architectures, to increase classification accuracy, especially in specialized tasks involving parking and traffic congestion. By combining sophisticated deep learning models, image processing, object detection frameworks like YOLO and attention mechanisms, future research can produce more accurate, flexible, and reliable methods for identifying parking occupancy and traffic congestion in smart cities. This chapter emphasizes how continuous innovation is required to create precise, customized models that fully utilize ITS's potential.

Chapter 3 : Parking space detection using deep learning

3.1 Introduction

Urbanization and the increasing number of vehicles in urban areas have intensified the demand for efficient parking space management. This challenge contributes to traffic congestion, elevated carbon emissions, and inefficient use of available spaces, leading to heightened stress for drivers. In many parking lots, floor sensors are used to determine the status of different parking spaces. This requires installation and maintenance of the sensors in each parking space, which could be expensive, especially in car parks with a high number of places available. To remedy these problems, researchers are oriented to propose using video cameras to detect the occupancy of vehicle parking lots. However, despite these efforts, detection of vacant parking spaces using only visual information remains an open problem. To address these concerns, researchers have turned to artificial intelligence (AI), particularly deep learning, to develop light and accurate models capable of parking space occupancy.

3.2 The proposed solution

In the solution, we propose two approaches for parking detection [168,169]. The first leverages lightweight convolutional neural networks (CNNs) based on AlexNet model with dilated convolutions, while the second employs transfer learning with the Inception V3 architecture enhanced by image data augmentation. Our models aim to improve classification accuracy while reducing computational complexity, offering scalable and efficient solutions for parking management. The study utilizes the PKLot dataset, a widely recognized benchmark for parking space classification, to evaluate the proposed methodologies. The following sections further explore experimental results, limitations, and future directions, underscoring the potential of these approaches to enhance computational efficiency and accuracy in parking lot detection.

3.2.1 Costumized CNN for classifying roadside images parking spaces

In this section, we proposed a classification system of images associated to the parking spaces for vehicles on a roadside based on the convolutional neuron networks (CNN). The image is supposed to be captured by a camera installed on a roadside and sent to a central computer for processing. The treatment is followed by a classification based on DP (deep learning) to determine the occupation of the parking spaces. The proposed classifier model uses fewer calculation parameters that involved a faster model with applying the principle of dilation.

The proposed model has been evaluated and compared with the previous models using PKlot dataset, a set of well-known images publicly available. The evaluation results show the reliability of our proposal comparing with the previous works. The proposed solution consists of designing a real-time monitoring system for roadside parking using smart camera and providing an overall view on the roadside parking spaces. This system has two main steps, the first step is performed by the smart camera. It consists for extracting the associated images for each parking space from the captured image. This step is described in the following sections 3.2.2.1 The second classifies the images to determine the state of the parking spaces (free or occupied). This step is the main goal of our work which will be detailed in section 3.2.2.4. The result of the classification obtained will be sent to a server for later exploitation by the users.

3.2.2 Extracting captured images

The procedure of acquisition and pre-processing of the image captured by the camera is inspired by the work [67]. It is assumed that the edge of the road has C parking space distributed along the road. Each parking space is characterized by a center Xc, a rotation angle dc, a length hc and a width w_c with $c = 1 \ldots C$. We associate for each parking space a square mask of min size (hc, wc). By respecting the center thus defined X_c, the mask is projected on the entire image for an extraction of all the images associated with the parking space. Figure 3.1 shows the supposed image that captured by the camera.

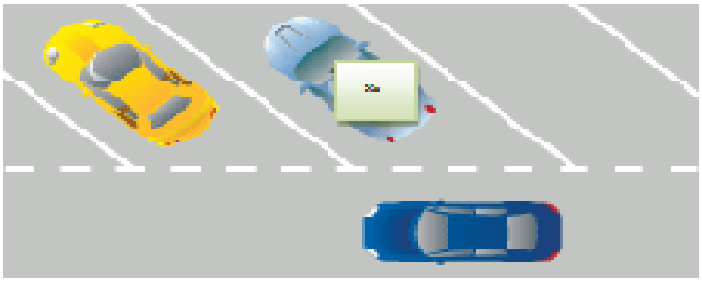


Figure 3.1: A region centered by X_c for the extraction of the parking space.

3.2.1.2 The AlexNet convolutional network

In order to propose a powerful model, we use AlexNet architecture [82] and we exploit the characteristics of the CarNet model [101]. CarNet is conceived in five layers of convolution; each layer is followed by a maximum clustering layer. For the high level of the model, two fully connected layers are added and followed by a final modal layer for a final classification into 1000 different classes. AlexNet model has been proposed for more complex visual recognition tasks than our binary classification problem. It was trained on an ImageNet LSVRC-2010 image database containing 1.2 million high resolution images to classify them into 1000 different

classes [62]. It showed a high-level performance for this task. For our case, this architecture is too deep for a binary classification (two classes) and it is therefore possible to create a more robust architecture concentrating on the number of dilated convolution layers.

3.2.1.3 Dilation concept for convolution

Diluted convolution is a means of exponentially increasing the receiving view (global view) by piling more and more dilated convolutional layers. This permets to widen the perceptual field without increasing the number of parameters or the quality of calculation. In simple terms, dilated convolution is simply a convolution applied to the capture with defined deficiencies. With this definition, the input being a 2D image, the dilation rate $k = 1$ is a normal convolution and $k = 2$ means ignore one pixel per input and $k = 3$, ignore 2 pixels. The figure 3.2 [101] shows a dilated convolution with different parameters.

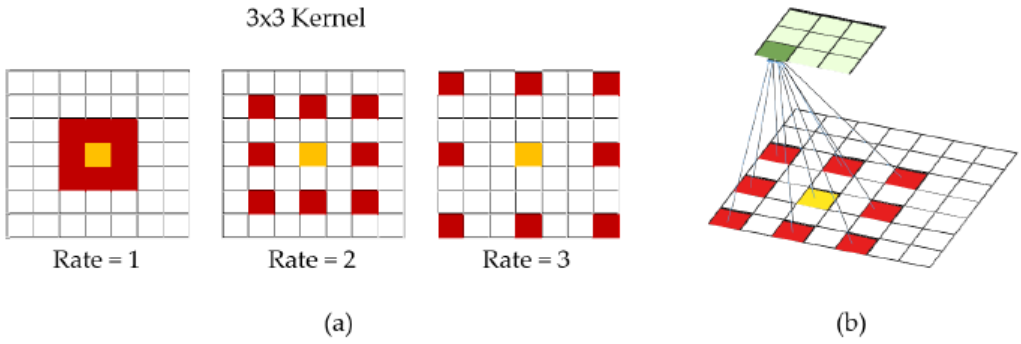


Figure 3.2 : (a) Examples of dilated convolutions. When the dilation rate is 1, it becomes a normal convolution. (b) Three-dimensional appearance of the dilated convolution (rate = 2).

Convolution dilation is used in applications to integrate knowledge into a larger context with lower cost. Furthermore, a model using dilated convolution runs faster than a model using normal convolution. This is where the idea of using dilated convolutions comes from to reduce the learning time of the model.

A good proposal presented by [152], which is applied for segmentation and action detection. The dilated convolution is also applied in areas other than vision. A good example is in [153] which presents a speech synthesis solution and in [154] the learning of text translation on time. They both use a dilated convolution to capture an overall view of the input with fewer parameters.

3.2.1.4 Structure of our model

The proposed model is based on AlexNet [82], where we remove the fourth convolutional layer. All convolutional layers conv1-3 with linear rectification (ReLU) followed by maximum

pooling. The number of conv1-3 filters is the number of neurons in fc4 and fc5 are drastically reduced to match the dimension of the problem. Our model contains the main points [168]:

- A dilation parameter is associated with each convolution layer with a rate that increases one step from the first layer to the third layer.
- The number of filters is fixed at 64 for the first layer and it will be increased layer by layer. The number of filters for the second layer is 128 and finally 256 filters for the last layer.
- Two fully connected has been added with fewer number of nodes.

The details of the proposed architecture are reported in Table 3.1.

Table 3.1: Architecture of our model.

Net	Conv1	Conv2	Conv3	Fc4	Fc5
our model	64x54*32 Conv 3x3 Pool 2x2 ReLU Dilation=1	128x27*16 Conv 3x3 Pool 2x2 ReLU Dilation=2	256x13x8 Conv 3x3 Pool 2x2 ReLU Dilation=3	1024 ReLU	2 Segmoid

The structure of our model consists of three convolutional layers each layer uses a 3x3 size filter. For each convolution layer, we apply dilation technique with a rate which varies gradually. We start with a standard dilation equal to 1 for the first layer, and then it increases by one step for the rest of the layers, which gives 2 for the second and 3 for the third layer. This indicates the difference between the present work and [101] which sets the dilation parameter to 2 for all convolution layers. Two fully connected layers have been added. The number of units for the two fully connected is 1024, which shows fewer parameter numbers comparing [101] . For the last layer that represents the output of the model, it has two units, since our goal is a binary classification. The three convolutional layers were provided with a linear rectification (ReLU) with a maximum pooling process (Max-Pooling). The number of filters is fixed at 64 for the first layer and it will be increased layer by layer. The number of filters for the second layer is 128 and finally 256 filters for the last layer. To avoid an overfitting of the model, Dropout regularization was used for fully connected layer and standard batch moralization has been used for each convolution layer. The proposed model accepts input images in 54x32 RGB format, like has been retained by the work [101]. To expand the size of our training dataset, we used the Keras deep learning library [155] that provides the ability to use data augmentation automatically when training a model. The data augmentation techniques can create variations of the images that can improve the ability of the fit model to generalize

what they have learned to new images. The training of our model on more data results in more skillful models. Learning the model takes about 8 hours for 75 iterations on a machine I3 (2.4 GHZ) of 4 GRAM using the CPU mode. The batch size is 100 and the number of epochs is 50. To optimize our training, we use Stochastic Gradient Descent algorithm with learning rate = 0.0001, weight decay= 0.0004 and momentum = 0.99.

3.3 Experimental and discussion

3.3.1 Performance evaluation based on accuracy

This chapter employs accuracy as the primary metric to assess the performance of the parking lot occupancy classifier, calculated using the standard definitions of True Positives (TP), False Positives (FP), True Negatives (TN), and False Negatives (FN).

- TP: The number of samples that the model correctly predicted as positive.
- FN: The number of samples that the model incorrectly predicted as negative.
- TN: The number of samples that the model correctly predicted as negative.
- FP: The number of samples that the model incorrectly classified as positive.

By measuring the ratio of correctly classified samples to the total number of samples, this statistic assesses the overall accuracy of the classification. Equation (3.1) defines it mathematically:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.1}$$

3.3.2 Dataset

To validate and test the performance of the proposed model, we use PKLot dataset [49]. Where the images taken from the cameras were treated to generate parking space images. In our experimental, the images are organized into three subsets by UFPR04, UFPR05 and PUCPR and it conducted for nine experiences as mentioned in Table 3.2.

3.3.3 Experimental results

In Figure 3.3, we evaluate the accuracy and loss of our proposed model. These metrics are evaluated using the three PKLot subsets, PUCPR (Figure 3.3 (a)), UFPR04 (Figure 3.3 (b)) and UFPR05 (Figure 3.3 (c)). For each experiment, we compare the accuracy and loss with the training phase. We use a part of each subset for training and we utilize the rest part for validation. We divided each subset of the dataset into five parts and 80% is used for training while 20% for validation. In addition, shuffling was performed after each epoch.

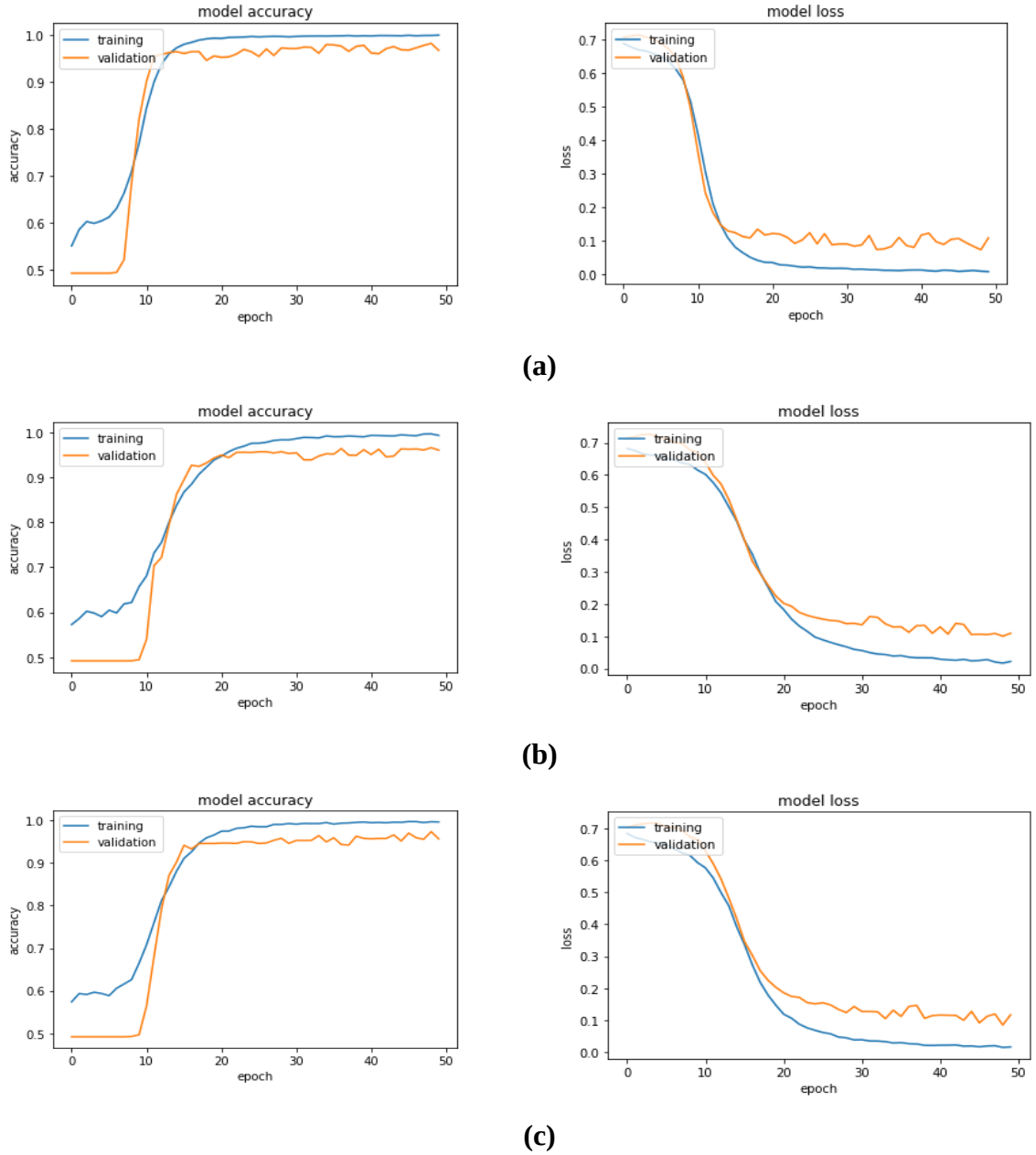


Figure 3.3 : (a) PUCPR, (b) UFPR04, (c) UFPR05.

We observe that training and validation accuracy increases with the increasing of number of epochs. Epoch represents the number of part on which the data is divided. When data is too big, we divide it into smaller sizes and give it to our computer one by one and update the weights of the neural networks at the end of every step to fit it to the data given. We observe that the PUCPR subset gives highest performance (high accuracy and low loss). Table 3.2 shows the accuracy values of the experimental results.

Table 3.2: Accuracy values.

Dataset	Training	Testing	Training Accuracy (%)	Accuracy (%)
PUCPR	PUCPR	PUCPR	99.91	98.94
	PUCPR	UFPR04	99.88	96.59
	PUCPR	UFPR05	99.81	98.99
UFPR04	UFPR04	UFPR05	99.34	98.27
	UFPR04	PUCPR	98.45	94.75
	UFPR04	UFPR04	99.25	94.92
UFPR05	UFPR05	UFPR05	99.00	92.89
	UFPR05	PUCPR	99.36	95.98
	UFPR05	UFPR04	99.36	96.92

Overall, we conducted nine types of experiments. we used 50 training epochs in our experiments. The highest results were obtained for the PUCPR subset, which implies that the PUCPR set is the least difficult. It is clear that the subsets UFPR04 and UFPR05 are getting closer, which means that the two subsets are getting closer in terms of complexity. In the nine experiments, our model presents some interesting results in terms of accuracy, 98.94%, 98.27 and 92.89 for PUCPR, UFPR04 and UFPR05 respectively. In particular, with the dilation technique, our model is faster in time of learning than a model which does not use the dilation. Reducing the number of nodes for the two fully connected layers of our model with the value 1024 lower than that of the CarNet model [101] produces a model with fewer parameters, which gives a very adaptable model to be embedded in a camera intelligent. Moreover, UFPR04 subset shows the lowest achievements. This subset contains the most difficult images, which contain more obstacles, shadows, trees, etc. For more precision and comparison, Table 3 displays the score values of this experiment.

We compared the results obtained by our model with the CarNet model [101]. This comparison was made on the PKLot dataset. The accuracy and number of calculation parameters for each model are shown in Table 3.3.

Table 3.3: CarNet [101] and our model comparison.

Model name	Training subset	Testing subset	Accuracy (%)	Mean (%)	Number of Trainable parameters
CarNet [101]	UFPR04	UFPR04	95.6	97.04	65.725.121
	UFPR04	UFPR05	97.6		
	UFPR04	PUCPR	98.3		
	UFPR05	UFPR04	95.2		
	UFPR05	UFPR05	97.5		
	UFPR05	PUCPR	98.4		
	PUCPR	UFPR04	94.4		
	PUCPR	UFPR05	97.6		
	PUCPR	PUCPR	98.8		
Our model [168]	UFPR04	UFPR04	98.27	**96.64**	**6.666.217**
	UFPR04	UFPR05	94.92		
	UFPR04	PUCPR	94.75		
	UFPR05	UFPR04	96.92		
	UFPR05	UFPR05	92.89		
	UFPR05	PUCPR	95.98		
	PUCPR	UFPR04	96.59		
	PUCPR	UFPR05	98.99		
	PUCPR	PUCPR	98.94		

Comparing CarNet accuray results with our model, we find that for some subsets, the CarNet model is more efficient for our model and conversely for others. Moderately, the CarNet model is more efficient for our model. The miniaturization and embedding factor always require optimized programs for good consumption of material resources. Our model clearly presents this need because it has fewer computational parameters than CarNet and uses a progressive dilation value greater than that of CarNet. This makes our model more computationally efficient.

3.4 Transfer Learning for Parking Occupancy Detection

3.4.1 The proposed model

Our model is created using transfer learning based on the Inception V3 model [156]. To minimize the size of the data, the Inception module utilizes different sizes of filters and maximum pooling. The benefit of this strategy is that it results in richer features with substantially less computation and lower parameters [157].

A pre-trained Inception V3 architecture served as the foundation for our model. To determine parking lot occupancy, a custom binary classifier was used in place of the original final layer. The final five layers of the Inception V3 network were frozen in order to maintain the pre-

trained feature extraction capabilities. In addition, we added a GlobalAveragePooling2D layer, two dropout layers with a rate of 0.25, two dense layers were activated by the relu function. Batch Norm, a normalization technique applied between a neural network model's two dense layers, has been used to speed up training, use faster learning rates, and make learning easier. The model is terminated by final dense layer with a single unit, activated by a sigmoid function. The proposed model is shown in figure 3.4. Finally, we have refined the model by SGD optimizer with a very low learning rate (0.0001) in order not to quickly disturb the weights which are already relatively well optimized. Our transfer learning model is trained for 50 epochs.

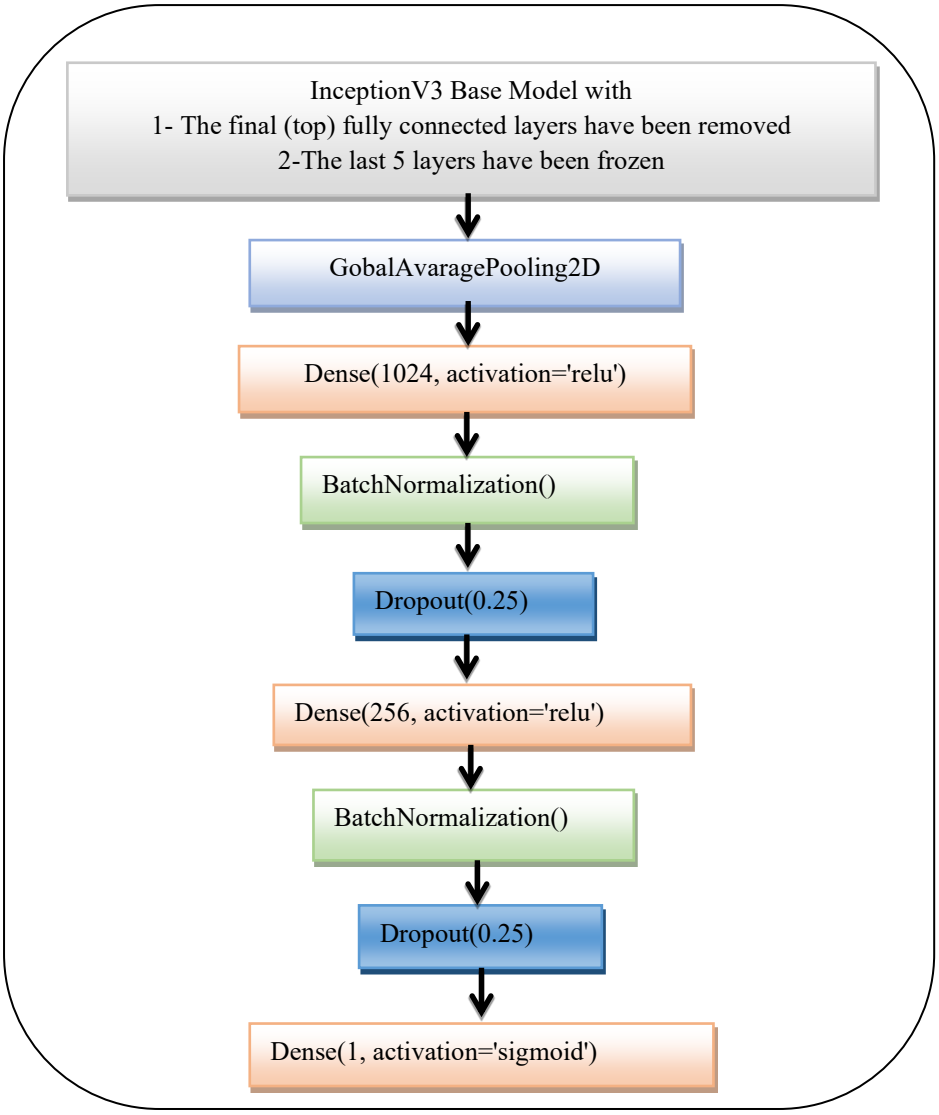


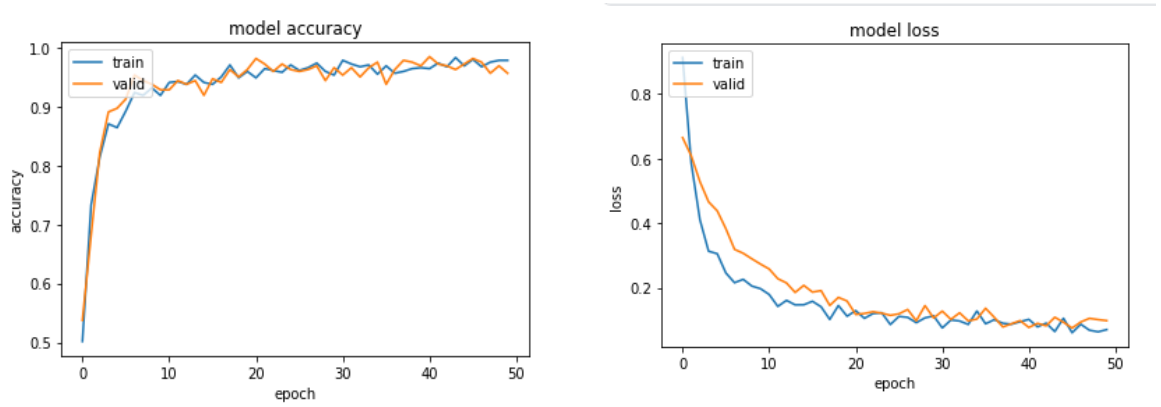
Figure 3.4 : The proposed model architecture.

3.4.2 Image data augmentation

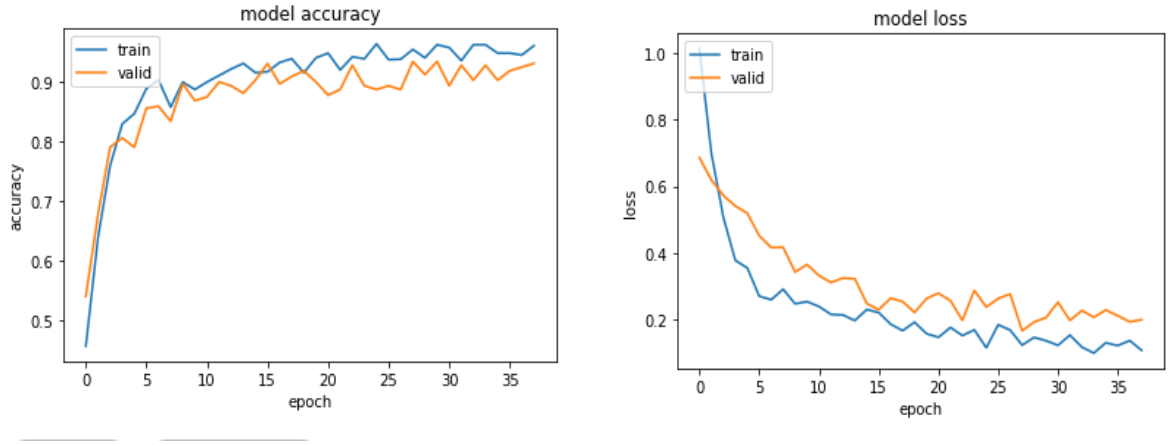
In deep learning, increasing the size of the dataset can significantly improve the accuracy of the CNN model [158], for this reason, in our case, we used a sub-set of PKLot in which their size was increased by applying the data augmentation algorithms. The image data augmentation technique generates different versions of the real image dataset by executing simple transformations such as horizontal flip, color space augmentation, random, cropping, brightness, translation, rotation, and scaling. In our case, we randomly return the inputs horizontally, we define the shear intensity and the range for the random zoom respectively with the range of 0.2 and 10.

3.4.3 Experimental results

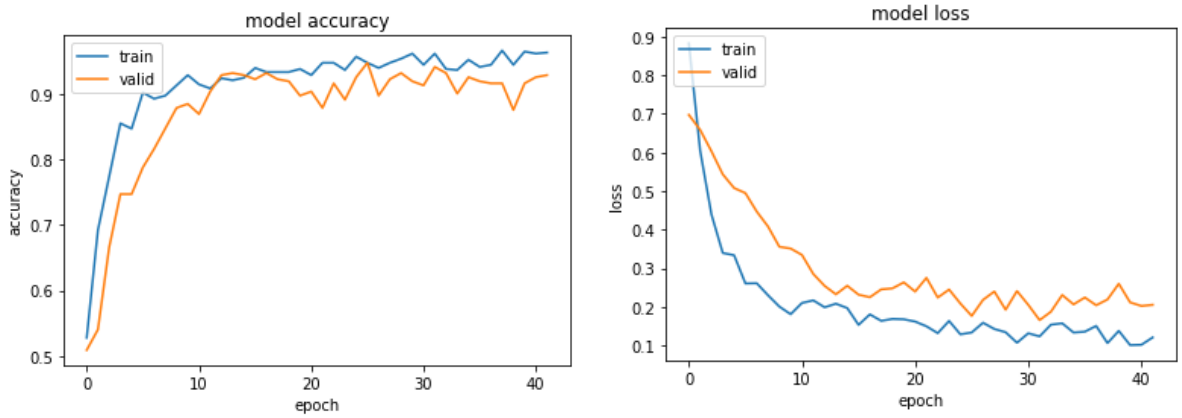
The evaluation of our model is shown in figure 3.5. We use accuracy as defined in Section 3.3.1 and loss as evaluation metrics. We built three subsets: PUCPR, UFPR04 and UFPR05. Each subset is divided into two subsets (training and validation) .Training of the model consumes 50 epochs with subset of data and tested by this same subset. The model's training and validation variation curve is presented by the six graphs for the three subsets (PUCPR(Figure 3.5(a)), UFPR04(Figure 3.5(b)), UFPR05 (Figure 3.5(c)). During the 50 learning iterations, figure 3.5 shows that the learning accuracy increases with the same value as the validation accuracy. Even for the validation loss, it decreases with the same value as the learning loss. This indicates that our model does not suffer from overfitting, a situation in which our model does not predict a learning error.



(a)



(b)



(c)

Figure 3.5: Different subsets, (a) PUCPR, (b) UFPR04, (c) UFPR05. Training and validation. x-direction: number of epochs (50), on the left y-direction: accuracy; on the right y-direction: loss

The experiments we conducted used 50 training epochs. 27 different types of experiments were used to train the model, and we created every possible combination between the subsets (PUCPR,UFPR04 and UFPR05). In the 27 experiments, our model shows interesting results in

terms of accuracy, 98.6%, 97.58 and 97.08 for PUCPR, UFPR04 and UFPR05 respectively. Table 3.4 lists all possible combinations along with their accuracy results.

Table 3.4 : accuracy values

Dataset	**Training**	**Testing**	**Training Accuracy (%)**	**Mean (%)**
PUCPR	PUCPR	PUCPR	97.27	92.99
		UFPR04	86.84	
		UFPR05	83.55	
	UFPR04	PUCPR	98.09	
		UFPR04	88.05	
		UFPR05	83.9	
	UFPR05	PUCPR	97.84	
		UFPR04	85.84	
		UFPR05	85.6	
UFPR04	PUCPR	PUCPR	96.64	
		UFPR04	97.58	
		UFPR05	91.56	
	UFPR04	PUCPR	94.13	
		UFPR04	97.48	
		UFPR05	91.06	
	UFPR05	PUCPR	96.44	
		UFPR04	97.28	
		UFPR05	89.05	
UFPR05	PUCPR	PUCPR	93.78	
		UFPR04	97.08	
		UFPR05	94.47	
	UFPR04	PUCPR	94.38	
		UFPR04	95.88	
		UFPR05	94.67	
	UFPR05	PUCPR	91.78	
		UFPR04	94.97	
		UFPR05	95.58	

We found that the best result was obtained for the PUCPR subset. This result reflects the simplicity of the PUCPR set in terms of the visual content of the image. The images of the UFPR04 and UFPR05 are less simple by comparing with PUCPR which interpret the value of accuracy is lower than that of PUCPR. The lowest accuracy for the UFPR04 and UFPR05 in comparing to the PUCPR and the UFPR05 is justified by the difficult subsets. The images of the UFPR04 and UFPR05 images are complex which contain more obstacles, shadows, trees, etc. Since transfer learning is the foundation of our model, only other works that employ the same methodology are compared. A pertinent benchmark for assessment was provided by two recent studies [98,92] that also used transfer learning for parking lot image classification.

A crucial trade-off between achieved accuracy and parameter efficiency is revealed by the comparative analysis of model performance, which is presented in Table 3.5. Although the VGG16 architecture [98] uses a significant 183 million parameters to achieve perfect accuracy (97.5%), this comes at the cost of computational complexity. Comparably, the mAlexNet model [92] still needs 31 million parameters despite achieving a high accuracy of 90.1%. On the other hand, the suggested model, which is based on a modified Inception V3 architecture, exhibits superior parameter efficiency, using only 24.4 million parameters to achieve a competitive accuracy of 92.99%. This is roughly 13% of the parameters needed by VGG16 and 79% of those used by mAlexNet. This outcome is important because it shows that our architecture effectively uses Inception's effective design principles (like factorized convolutions) to significantly reduce model size and computational footprint without sacrificing predictive power. Because of the significant efficiency gains and the slight loss of absolute accuracy, the model is now a more viable option for real-world applications where storage, processing power, and inference speed are crucial limitations. Future work will focus on closing the remaining accuracy gap through techniques like advanced data augmentation and hyperparameter optimization without compromising this efficiency.

Table 3.5: comparison of our model with [98] and [92]

Model reference	Network used	Accuracy (%)	Number of parameters
[98]	VGG16	95.7	183 million
[92]	mAlexNet	90.1	31 million
Our model [169]	Inception V3	92.99	**24.432.417**

3.5 Conclusion

In this chapter, we have explored and developed two efficient deep learning models for detecting vehicle parking spaces, contributing to the advancement of intelligent transportation systems in smart cities.

The first approach introduced a robust CNN architecture, optimized through dilated convolution layers with a gradual increase in dilation rates from the first to the third convolutional layer. Additionally, by reducing the number of nodes in the fully connected layers, we achieved a lightweight model with fewer training parameters. This optimization not only improved computational efficiency but also facilitated deployment on smart cameras such as Raspberry Pi for real-time execution.

The second approach leveraged transfer learning based on Inception V3, where the pre-trained model was refined by removing the final fully connected layers and integrating a GlobalAveragePooling2D layer, two dropout layers, and three dense layers. Data augmentation

was applied to enhance the model's generalization capability, leading to promising accuracy in detecting parking spaces.

The parameter efficiency of the two models presented in this chapter is a major benefit; they have a lot fewer parameters than works in the context of transfer learning. In optimization-focused contexts where resource constraints are a top concern, they are perfect for delivering dependable services because of their efficient architecture, which also lowers computational overhead and increases their robustness.

The integration of these models into smart city infrastructures offers significant benefits, including reduced traffic congestion and lower CO_2 emissions by assisting drivers in locating available roadside parking spaces efficiently. Future work will focus on further improving model performance through experimentation with additional datasets, optimizing configuration parameters, and conducting real-world deployments for enhanced robustness. Ultimately, deploying our model on smart cameras for real-time monitoring of vacant and occupied parking spaces will contribute to the development of more sustainable and intelligent urban mobility solutions.

Chapter 4 : Vision transformer for detecting traffic congestion

4.1 Introduction

Severe traffic congestion brought on by the fast urbanization and rise in automobile ownership has resulted in lost revenue, pollution, and longer travel times. The development of intelligent transportation systems (ITS) that can optimize traffic flow and minimize delays depends on the efficient detection and classification of traffic congestion. Conventional approaches depend on loop detectors and traditional image processing methods, which frequently fall short in the face of real-world complexity like occlusions and changing weather.

In image recognition tasks, recent developments in intention mechanisms, especially Vision Transformers (ViTs), have demonstrated considerable promise. Despite these developments, current models frequently have issues with computational efficiency, real-time applicability, and traffic condition adaptability. In order to fill these gaps, we incorporate YOLOv8s for vehicle detection before transferring the mapped images to a lightweight UniViT model with single-head attention for effective traffic congestion classification. The suggested method is presented in this chapter and tested on the UCSD dataset, which comprises a variety of weather conditions, including clear, cloudy, and rainy scenarios.

4.2 Methodology

Our method hinges on utilizing a vision transformer to analyze road traffic status based on images. Before delving into the intricacies of this transformer, we outline how its inputs are prepared. As illustrated in Figure 4.1, our comprehensive method [170] is structured into three distinct stages, each of which is individually tailored. These stages are elaborated upon in the subsections below.

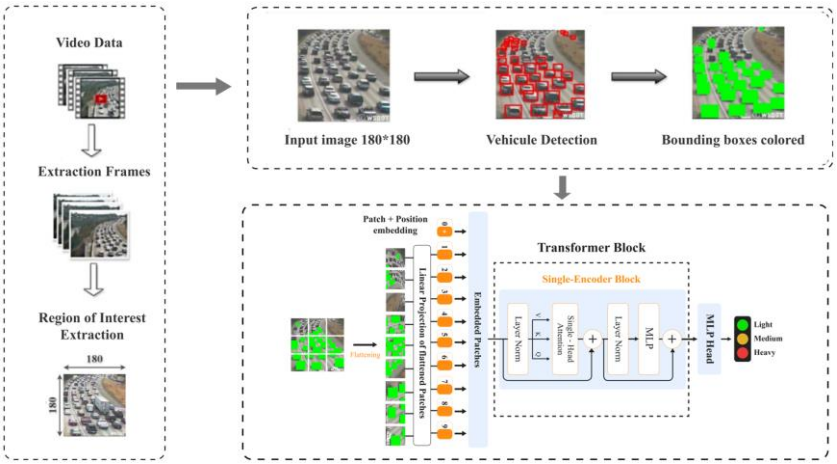


Figure 4.1: The proposed method, highlighting the stages involved in traffic congestion classification.

4.2.1 Dataset preparation

We conducted an evaluation of our approach and compared its performance with state-of- the-art methods, utilizing a publicly available dataset that has been extensively used in prior research studies. The dataset used is UCSD (University of California San Diego), a publicly available dataset [151], which described in chapter 2. The videos in this dataset have a duration of about 5 seconds and it displays a frame rate that varies between 42 to 52 frames recorded at 10 frames per second (fps) with a 320 x 240 pixels resolution. We extracted frames that then underwent data cleaning, during which we manually inspected each image to remove any faulty ones, typically found at the beginning of each video. We also eliminated blurred, ambiguous, or unclear samples, as well as those affected by obstacles such as water droplets on the camera lens or recording issues. This cleaning process ensures that the dataset consists of high-quality, reliable images, thereby creating a strong foundation for the subsequent steps of our proposed approach and enabling accurate traffic congestion classification. Manual multi-classification is used to categorize sample images into three classes: 'Heavy', 'Medium' and 'Light', denoting heavy, medium, and low congestion, respectively. In total, we obtained 10111 images classified into 1851 frames of heavy traffic, 2108 frames of medium traffic, and 6152 frames of light traffic. Additionally, we defined a Region of Interest (ROI) to exclude irrelevant external objects and focus on the road congestion area. By selecting the lower corner of the image, we concentrated on the most crucial part for congestion detection. The original images had a size of 320x240, and with the ROI selection, they were resized to 180x180. This technique reduced the computational complexity, prevented memory issues, and improved classification accuracy by focusing solely on congestion-related features. The main purpose of this study was to classify traffic congestion into three classes. Table 4.1 shows the details of splitting dataset.

Table 4.1:The distribution of the image dataset for training, validation and testing.

Class	Training	Validation	Testing
Heavy	1251	300	300
Medium	1508	300	300
Light	5552	300	300
Total	8311	900	900

4.2.2 Image mapping

All Vehicle detection stands as the pivotal element in this congestion detection method, with the objective of accurately quantifying the number of vehicles within a specified road segment. Achieving precise vehicle detection is essential for the overall efficacy of the proposed method. To this end, a range of sophisticated techniques has been developed including frame differencing [135], optical [159], Region-based Convolutional Neural Network (R-CNN) [160],

and You Only Look Once (YOLO) [161]. The authors [162] investigated for vehicle detection methods Gaussian Mixture Model (GMM), GMM-Kalman filter, Optical Flow, and Aggregate Channel Features (ACF) object detector in both urban and highway settings. Moreover, a comparative analysis facilitates the selection of the most appropriate methods for integration to address traffic congestion issues.

Image mapping stage consists on simplifying the dataset, aiming to focus exclusively on detecting bounding boxes rather than the details of the vehicles. In our method before training our proposed UniViT model for traffic congestion classification, we utilized the Yolov8s model from [163] to identify objects within the input images, offering superior performance and accuracy while maintaining high inference speeds and it has already trained on a large-scale dataset, such as COCO dataset [164]. In the process of vehicle detection, the aim is to discern each vehicle individually, marking them out by enclosing them within rectangular boxes. These boxes, once established, are then colored in green. This method allows for clear and precise identification of vehicles within the captured imagery, facilitating subsequent analysis and processing tasks. This decreases the computation time and reduce the complexity of the task.

The selection of the optimal object detector for our car detection task, we conducted a comparative analysis of YOLOv5s, YOLOv7, and YOLOv8s. We leveraged official metrics (mAP and speed) reported on the COCO dataset using an A100 GPU as a baseline for performance evaluation [165]. To assess model suitability for our specific use case, we constructed a representative dataset comprising 50 images per class for a total of three classes. Manual annotations were provided to determine the ground truth for car counts. Each YOLO model was then employed to detect cars within these images, and the resulting detections were compared against the ground truth to calculate task-specific mAP and average inference speed. Based on this comprehensive evaluation, YOLOv8s emerged as the preferred choice due to its superior balance of accuracy and speed when applied to our dataset. Table 2 represent the performance analysis between the different object detectors.

Table 4.2: The performance analysis of YOLO v5, v7, v8

Object detector	**Average Precision (%)**		**Speed (ms)**	
	Coco dataset [165]	**UCSD dataset**	**Coco dataset [165]**	**UCSD dataset**
YOLO v5s	43	44.12	1,27	132.73
YOLO v7	51.4	57.27	2.8	604.87
YOLO v8s	53.78	53.78	1.2	176.81

4.2.3 UniViT model

Our proposed UniViT model as showing in Figure 4.2. The transformer encoder comprises layer normalization, single-head attention, and a multi-layer perceptron block. Utilizing single-head attention offers the advantages of reducing the complexity of the model, thereby enhancing computational efficiency and facilitating deployment in resource-constrained environments. The normalization layer is applied to normalize the input data, reducing the training time and enhancing training stability.

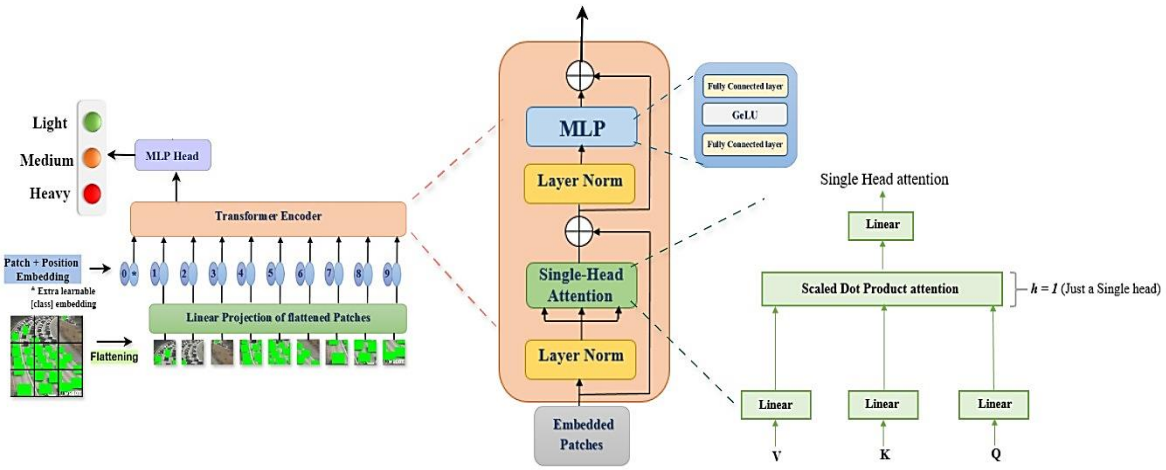


Figure 4.2 : The architecture of the UniVit.

4.2.3.1 Single-head self-attention

The self-attention mechanism is pivotal in capturing dependencies across different parts of an input sequence by allowing each element to dynamically focus on other elements within the same sequence. In a single self-attention mechanism, each element in the input sequence is projected into three distinct vectors: the Query (Q), Key (K), and Value (V) vectors with the simplification that only one set of Query, Key, and Value vectors is used. These projections are achieved through learned linear transformations. Formally, given an input sequence of tokens $X=\{x_1,x_2,\ldots,x_n\}$ each element x_i is projected into Query (Q), Key (K), and Value (V) vectors using learned linear transformations, , as described by equations (4.1- 4.3):

$$Q = X.W_Q \tag{4.1}$$

$$K = X.W_K \tag{4.2}$$

$$V = X.W_V \tag{4.3}$$

where W_Q, W_K and W_V are the learned weight matrices. The dimensions of W_Q, W_K and W_V are typically $d_x \times d_q$, $d_x \times d_k$ and $d_x \times d_v$ where d_x is the input feature dimension, d_q, d_k and d_v are the dimensions of the key vectors. The attention scores are calculated by taking the dot product of the query vector for each element with the key vectors of all elements. This produces a score matrix as presented by equation (4.4):

$$\text{Attention Score} = \left(\frac{QK^T}{\sqrt{d_K}}\right) \tag{4.4}$$

Where Q and K are matrices containing the query and key vectors for all elements in the sequence, and $\sqrt{d_K}$ is a scaling factor to counteract the effect of large dot-product values and ensure numerical stability. The attention scores are passed through a Softmax function to obtain the attention weights. This step normalizes the scores into a probability distribution, as given by equation (4.5):

$$\text{Attention Weights} = \text{Softmax}\left(\frac{QK^T}{\sqrt{d_K}}\right) \tag{4.5}$$

The Softmax function is applied row-wise, ensuring that the attention weights for each query vector sum to one. The final output for each element, representing the output of the single-head self-attention, is obtained by computing the weighted sum of the value vectors, as presented by equation (4.6):

$$\text{Single Head Output} = \text{Attention Weights} \cdot \text{V} \tag{4.6}$$

4.2.3.2 Multilayer perceptron (MLP)

The transformer block integrates an MLP module following self-attention, which includes two fully-connected linear layers separated by a Gaussian Error Linear Unit (GELU) activation function, as specified in [166]. Dropout is incorporated to facilitate the model in learning complex representations. The structure of this module is depicted on the top right side in Figure 2. In our research, the MLP is composed of fully connected layers with hidden units configured as 128 and 64. This MLP architecture is purposefully crafted to capture intricate patterns and relationships within the encoded image representations.

4.2.3.3 Multilayer Perceptron Head

The proposed architecture diverges from traditional transformers as defined in [34] by omitting the decoder and incorporating the MLP head for final classification. The output of the transformer block is projected into a lower-dimensional space with three units, corresponding to the three traffic congestion classes: heavy, medium, and light. The class token from the transformer block is fed into this classification head, and the class with the highest probability is chosen as the predicted congestion level. In this study, the MLP head comprises fully connected layers with 2048 and 1024 hidden units, respectively.

4.2.3.4 Classification process of the UniVit

To classify traffic states (light, medium, and heavy) using the proposed UniViT model, the process consists of several key steps, outlined as follows:

- Input preparation:
  - ✓ Initially, the model accepts 2-dimensional RGB image data with a size of 180x180 pixels, representing traffic scenes with green-colored bounding boxes as a result of the image mapping method.
  - ✓ Patch embeddings: The UniViT splits each input image into small fixed-size patches of 16x16x3, where 3 represents the number of channels (RGB). Given our image size of 180x180 pixels, the number of patches will be 121, as calculated by $N = 11 \times 11 = 121$ patches per image. Each patch is flattened into 768-dimensional vectors and then linearly projected into a higher-dimensional space to create patch embeddings.

- Embedding and positional encoding:
  - ✓ positional encoding: Since transformers lack intrinsic positional information (unlike CNNs), a positional encoding is added to each patch embedding to provide spatial information to the model.

- Attention mechanism (Single-Head):
  - ✓ self-attention calculation: In a single-head attention mechanism, for each patch, you calculate the attention score using the Query, Key, and Value matrices derived from the patch embeddings.
  - ✓ attention weights: The attention weights determine the importance of each patch relative to the others. The weighted sum of the Value matrix provides an updated representation of the patches, factoring in the relationships between patches.

- Transformer block:
  - ✓ feed-forward layer: After the attention mechanism, the output goes through a feed-forward neural network. This is followed by normalization and residual connections to stabilize the learning process.
  - ✓ multiple block: Typically, the transformer block (witch consisting of self-attention and feed-forward layers) is stacked to capture complex relationships across patches.

- Classification token:
  - ✓ A special classification token is added to the input sequence of patch embeddings. After passing through the transformer block, this token captures a summary representation of the entire image.
- Output head:
  - ✓ single linear layer: the UniViT lacks a decoder, the classification is done using a simple linear layer connected to the classification token output from the final transformer block.
  - ✓ Output probabilities: The linear layer projects the classification token to a 3-dimensional output (representing the three traffic states: light, medium, and heavy).
  - ✓ Softmax activation: Apply a softmax function to the 3-dimensional output to get probabilities for each class.
- Classification:
  - ✓ Argmax for classification: The class with the highest probability is selected as the final prediction (light, medium, or heavy).

4.3 Experimental results

This section provides an analytical experiments and comprehensive discussion on the results obtained through the implementation of the proposed methodology. The implementation of UniVit is conducted using python and the keras package with tensorFlow. The computations were performed on an Intel(R) Core(TM) i5-11400 2,9 GHz processor with 16 GB of RAM. For the subsequent experiments, the selected hyperparameters for training the UniVit were fixed after multiple iterations, as shown in Table 4.3.

This section provides an analytical experiments and comprehensive discussion on the results obtained through the implementation of the proposed methodology. The implementation of UniVit is conducted using python and the keras package with tensorFlow. The computations were performed on an Intel(R) Core(TM) i5-11400 2,9 GHz processor with 16 GB of RAM. Table 4.3 shows the hyperparameter values used to train the UniVit model (indicated in bold). These hyperparameters were selected after conducting multiple experiments with the AdamW optimizer. L1 and L2 are regularizer objects utilized in our UniVit, serving as mechanisms to prevent overfitting and improve the model's generalization. In Table 4.3, the number of epochs is set to 100, as this value provided the best accuracy during several tests.

Table 4.3 : Hyperparameters tuning.

Learning rate	Weight decay	L1	L2	Dropout				Batch size	Training	Validation	Test
									Accuracy (%)		
0.001	0.0001	0.01	0.01	0.5	0.5	0.5	0.5	32	94.94	97.24	98.11
0.001	0.0001	0.001	0.001	0.3	0.3	0.3	0.3	32	98.01	99.15	99.78
0.001	0.0001	0.001	0.001	0.1	0.1	0.1	0.1	32	98.86	99.89	99.89
0.001	0.0001	0.001	0.001	0.1	0.1	0.1	0.1	256	99.78	99.33	99.33
0.001	0.0001	0.001	0.001	0.1	0.1	0.1	0.1	64	99.72	99.44	99.77
0.0001	0.0001	0.01	0.01	0.5	0.5	0.5	0.5	32	92.49	90.11	89.11
0.0001	0.0001	0.001	0.001	0.3	0.3	0.3	0.3	32	97.77	98.22	97.89
0.0001	0.0001	0.001	0.001	0.1	0.1	0.1	0.1	32	98.32	97.33	97.89
0.0001	**0.0001**	**0.001**	**0.001**	**0.1**	**0.1**	**0.1**	**0.1**	**32**	99.81	99.67	99.89

4.3.1 Evaluation metrics

Our evaluation of the UniViT model for traffic congestion classification utilizes several key metrics. These include accuracy, which follows the definition established in section 3.3.1, in addition to precision, recall, and the F1 score.

4.3.1.1 Precision

Quantifies the ability of the model to correctly identify positive instances out of all the instances classified as positive. It is calculated by dividing the number of true positives by the sum of true positives and false positives. It is defined by equation (4.7):

$$\text{Precision} = \frac{TP}{TP + FP} \tag{4.7}$$

4.3.1.2 Recall

Measures the ability of the model to identify all positive samples correctly. It is calculated as follows:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{4.8}$$

4.3.1.3 F1-score

The F1-score is the harmonic mean of precision and recall, providing a balanced measure that considers both metrics. Mathematically it is defined by equation (4.9):

$$\text{F1} - \text{score} = 2 \times \frac{Precision \times Recall}{Precision + Recal} \tag{4.9}$$

4.4 Performance evaluation of the UniViT

Typically, data imbalance poses a risk because the model may become overly focused on the majority class, leading to poor generalization for the minority classes. Common approaches to tackle this issue include oversampling the minority classes or undersampling the majority class.

In this work, the oversampling technique, which includes generating synthetic images, is employed in the dataset referenced in Table 4.1. This dataset is structured into three traffic congestion classes: Heavy, Medium, and Light. Each class contains 5,000 samples in the training set, resulting in a total of 15,000 training samples. For both validation and testing, 300 samples per class are provided, yielding 900 samples for each set. Conversely, the dataset is also structured using undersampling. In this case, it is divided into the same three traffic congestion classes: Heavy, Medium, and Light. Each class is represented by an equal number of samples in the training set, with 1,200 images per class, leading to a total of 3,600 training samples. Additionally, each class has 300 images allocated for both validation and testing, ensuring that the validation and testing sets are balanced, with 900 images each.

However, both techniques have significant downsides in our context. Instead, we chose to address the imbalanced dataset referenced in Table 1 by using evaluation metrics that are more robust to class distribution, such as the F1 score, which balances precision and recall and provides a clearer indication of the model's performance across all classes. Additionally, as shown in the confusion matrix, the model demonstrates no significant bias toward any specific class, confirming that the imbalanced data did not adversely affect overall classification performance.

4.4.1 Evaluation on balance UCSD dataset with oversampling

Figure 4.3 illustrates the training and validation accuracy (left) and loss (right) over the course of 30 epochs. The training accuracy starts at around 82% and gradually increases, reaching 98% by the final epoch. The validation accuracy follows a similar trend, starting at 86%, peaking slightly higher than the training accuracy at certain points, and ending close to 99%. For the loss curve, both training and validation losses experience a rapid decrease within the first 5 epochs, with the training loss dropping from over 6.0 to less than 0.1. The validation loss also reduces significantly, stabilizing around 0.1 after epoch 10. The validation loss remains slightly higher than the training loss throughout the process, yet both curves converge toward the end. The accuracy graph shows signs of overfitting, as the validation accuracy consistently exceeds the training accuracy at multiple points. This suggests that the model may be learning patterns specific to the validation data, potentially limiting its ability to generalize effectively to unseen data.

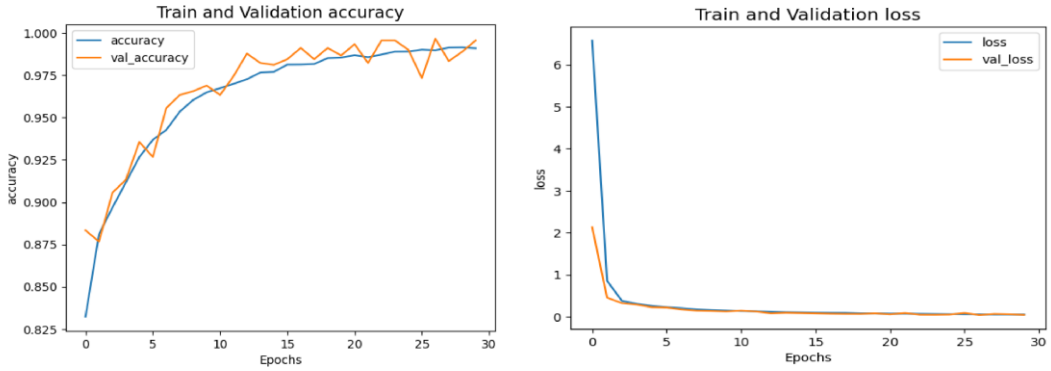


Figure 4.3 : Training and validation accuracy and loss graphs with oversampling.

4.4.2 Evaluation on balance UCSD dataset with underdamping

The graphs in Figure 4.4 present the evolution of training and validation accuracy (left) and loss (right) over 18 epochs. Initially, the training accuracy improves steadily, reaching approximately 95% by epoch 10, while the validation accuracy peaks slightly higher at around 97%. After epoch 10, the validation accuracy consistently remains higher than the training accuracy, which could indicate overfitting. By epoch 18, the validation accuracy stabilizes at 96%, while the training accuracy slightly lags behind at 94%.

Similarly, in the loss plot, the validation loss drops sharply during the first few epochs, stabilizing at around 0.15 by epoch 5 and remaining below the training loss throughout the training process. The training loss decreases more gradually, reaching approximately 0.20 by epoch 18. The fact that the validation loss is consistently lower than the training loss suggests that the model may be learning specific patterns from the validation set, potentially leading to overfitting.

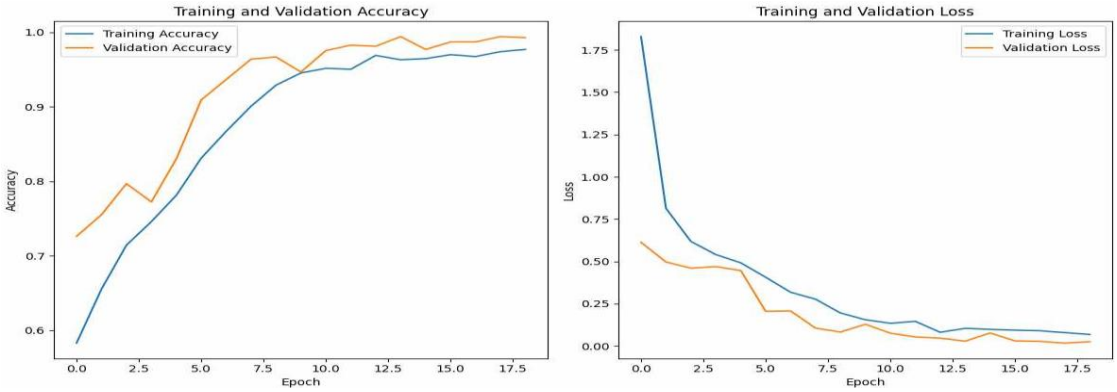


Figure 4.4 : Training and validation accuracy and loss graphs *with undersampling*.

4.4.3 Evaluation on imbalance UCSD dataset

Figure 4.5 illustrates the training and validation accuracy and loss graphs, showcasing the performance of the UniVit model over 100 epochs. The training accuracy rapidly increases and stabilizes close to 100%, while the validation accuracy follows a similar trend with slight fluctuations, indicating good generalization. Both training and validation loss decrease sharply initially and then stabilize. The close alignment of the training and validation metrics suggests effective learning. Despite training for 100 epochs, the model maintains stable performance. Overall, the model demonstrates efficient learning and generalization across the dataset.

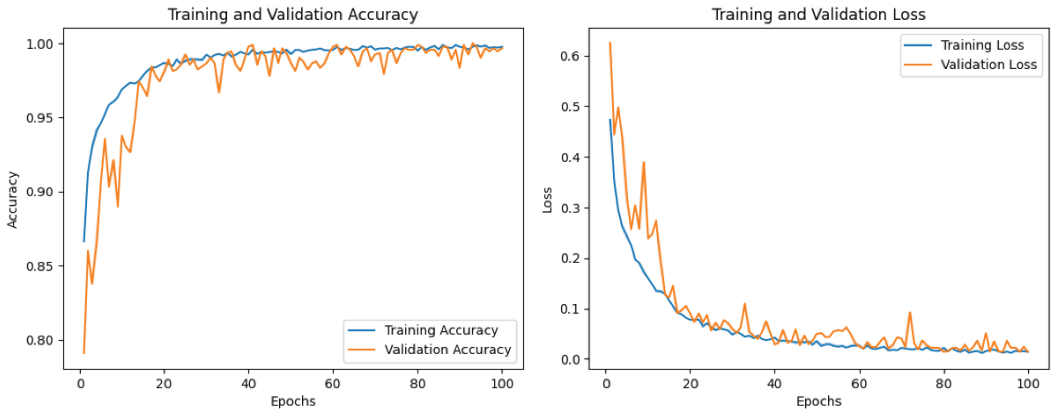


Figure 4.5: Training and validation accuracy and loss graphs.

The standard metrics for the proposed approach are displayed in Table 4.4. The obtained results, depict remarkable performance metrics with accuracy, precision, recall, and F1-score all hovering around 99.89%, demonstrating the exceptional predictive capabilities of the model. Such high values across these crucial metrics indicate the ability of the model to make highly accurate predictions, effectively distinguishing between positive and negative instances. This suggests that the model has been trained comprehensively and is adept at generalizing patterns within the dataset. The consistency of high performance across all metrics underscores the efficacy of the model in accurately capturing complex patterns and relationships within the data.

Table 4.4: Performance of the UniVit.

Metric	**Accuracy**	**Precision**	**F1-score**	**Recall**
Value	99.89%	99.89%	99.89%	99.89%

Figure 4.6 shows the confusion matrix for three classes (heavy, light, medium), demonstrating an exceptionally high-performing classifier. The matrix shows that the classifier achieves perfect classification for the heavy and light classes, with 100% of the true heavy instances correctly identified as heavy and 100% of the true light instances correctly identified as light. This means there are no misclassifications for these two classes, indicating that the classifier is

highly reliable and accurate for heavy and light categories. For the medium class, the classifier maintains a high level of accuracy, correctly classifying 99.67% of the true medium instances as medium. However, there is a slight misclassification rate of 0.33%, where medium instances are incorrectly classified as heavy. Despite this minor error, the classifier's performance for the medium class remains excellent, with a very high precision. Overall, the classifier's precision, recall, and F1-scores for all classes are extremely high, reflecting its robustness and reliability. Precision measures the accuracy of the positive predictions, while recall measures the completeness of the positive predictions. In this case, both metrics are near perfect for all classes, leading to an almost perfect F1-score, which is the harmonic mean of precision and recall.

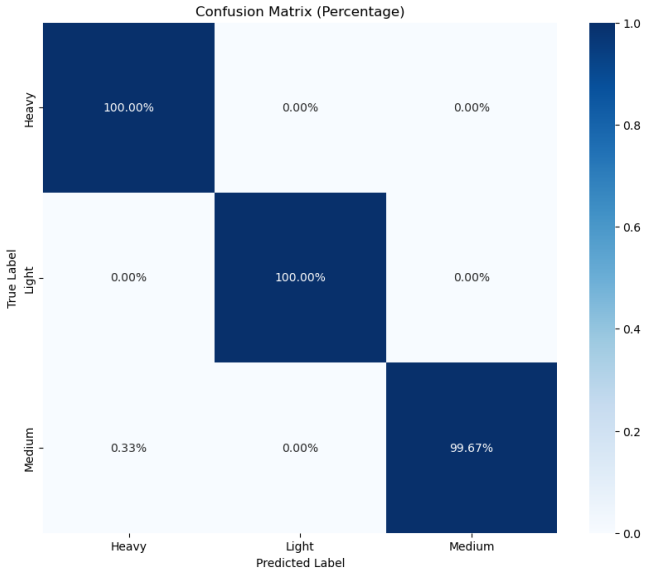


Figure 4.6 : Confusion matrix of the results of the UniViT model

The classifier's performance is commendable, especially considering the potential challenges in distinguishing between different levels of traffic congestion in images. The perfect classification for heavy and light congestion, along with the near-perfect classification for medium congestion, suggests that the classifier is well-trained and effectively captures the distinguishing features of each congestion level. This high level of accuracy is crucial for applications that depend on precise traffic congestion classification, as it minimizes errors and enhances the reliability of traffic management systems. The minor misclassification for medium congestion is negligible and does not significantly impact the overall performance, underscoring the classifier's robustness and effectiveness in handling the classification task.

4.4 Comparison of our method with others works

As shown in the Table 4.5, the comparative analysis of our proposed method with previous works demonstrates a significant advancement in accuracy for road traffic congestion classification.

Table 4.5: Comparison between the classification accuracies of our method with previous works.

Reference	Method	Image preprocessing employed	Using YOLO	Dataset	Accuracy (%)
[145]	Deep Convolutional Neural Network (DCNN)	Region of Interest (ROI)+Color-coding	YOLOv3	USCD	98.20
[150]	Vision Transformer	-	No	USCD	99.76
Our method [170]	Vision Transformer with a single head attention	Region of Interest (ROI) + Image mapping	YOLOv8	USCD	**99.89**

As shown in the above table, the comparative results show how parking space occupancy classification on the USCD dataset has advanced over time. When paired with strategic image preprocessing (ROI, color-coding) and object detection (YOLOv3), DCNNs are very effective for this task, as demonstrated by the high baseline accuracy of 98.20% set by [145]. The notable increase to 99.76% by [150] demonstrates the revolutionary potential of Vision Transformer (ViT) architectures, which are excellent at capturing global contextual relationships in an image without the need for explicit preprocessing. A new state-of-the-art accuracy of 99.89% is attained by our suggested technique, a Vision Transformer with a single-head attention mechanism. Two major innovations are responsible for this slight but significant improvement: first, the incorporation of a more sophisticated and effective object detector, YOLOv8, which offers higher-quality input regions; and second, the exploitation of YOLOv8 in the image mapping technique and conjunction with ROI extraction to further refine the input data.

Importantly, our model challenges the notion that higher model complexity is required for peak accuracy by achieving this superior performance with a simplified single-head attention architecture. With transformer-based models, this result highlights the significant influence of targeted preprocessing and contemporary detection frameworks, which together provide a more accurate and effective system for visual occupancy classification.

4.5 Conclusion

This chapter introduced a new method for classifying road traffic congestion using the YOLOv8s model to detect vehicles and a single-head attention mechanism transformer model. The method classifies road traffic into three categories: light, medium, and heavy. We tested its

performance against state-of-the-art methods using the publicly available UCSD dataset, which includes daytime road traffic videos recorded under various weather conditions (rainy, overcast, and clear). Our method achieved an accuracy of 99.89%, surpassing the best-known methods. The excellent performance of our method comes at a lower computational cost due to the image mapping stage, which simplifies the dataset by focusing on detecting bounding boxes rather than the finer details of the vehicles. Additionally, the architecture of UniViT, with its single head, enhances efficiency. When combined with YOLOv8s, UniViT demonstrates remarkable adaptability, making it easy to integrate into traffic management systems. This integration aims to improve traffic flow efficiency and reduce travel delays. Furthermore, it assists authorities in intelligent urban traffic planning by dynamically regulating traffic signals and redirecting traffic routes based on real-time congestion assessments.

Chapter 5 : Fuzzy logic for detecting traffic congestion

5.1 Introduction

Conventional traffic monitoring methods mainly rely on physical sensors like wireless sensor networks, radar systems, and inductive loop detectors. Despite their effectiveness, these approaches have drawbacks such as high installation and maintenance costs, limited scalability, and poor adaptability to changing urban growth. A viable substitute that provides constant visual monitoring of traffic conditions is video-based surveillance. This chapter addresses these issues by introducing a Multi-Module Road Traffic Estimation (MRTE) framework that detects traffic congestion by fusing fuzzy logic inference with lightweight image processing.

5.2 Multi-Module Road Traffic Estimation (MRTE)

MRTE [171] is made to work effectively in real time while retaining high accuracy, in contrast to deep learning models that need a lot of annotated data and a lot of processing power. It incorporates three supplementary modules:

- Measurement of vehicle density using Scale-Invariant Feature Transform (SIFT) descriptors.
- Traffic Velocity Estimation (TVE), which is based on changes in density values between frames.
- A fuzzy logic inference system is used to detect the congestion level.

MRTE provides a transparent decision-making process that captures the ambiguity and gradual changes present in actual traffic flow by utilizing fuzzy logic's interpretability. Because of this feature, the system is both computationally efficient and explicable, which is crucial for intelligent transportation decision-makers.

5.3 Network model and assumptions

We consider a Vehicular Ad Hoc Network (VANET) environment, in which roadside cameras are deployed to capture and transmit visual data reflecting road traffic conditions. These cameras are strategically placed along roadways to enable real-time capture of images or videos that depict traffic flow, environmental context, and roadway status. Leveraging VANET communication, this visual data can be efficiently transmitted to nearby vehicles or roadside infrastructure, supporting real-time traffic monitoring and management. Figure 5.1 illustrates the system of our environment. In this environment, each camera is assigned to cover a specific

road segment, denoted as S_i. The entire VANET is thus geographically partitioned, with each segment associated with a distinct camera deployment. Cameras are positioned based on various criteria, such as visibility range, coverage distance, and road curvature, to ensure optimal monitoring of highway segments with minimal overlap. The video feeds from these cameras serve as the primary input for assessing traffic conditions.

However, it is imperative to address the privacy and security issues associated with camera deployment in VANETs. This includes ensuring robust data protection measures, obtaining appropriate consent, and adhering to ethical guidelines. Furthermore, the deployment and maintenance of cameras in VANETs requires meticulous planning, including optimal positioning and routine maintenance, to ensure effective and efficient operation.

To formally model traffic flow, we define a directed graph $G = (V,E)$ be a directed graph representing road traffic, where $V = \{s_1, s_2, \ldots, s_n\}$. denotes a set of n road segments. Each segment $S_i = \{v_1, v_2, \ldots, v_m\}$ represents a set of m vehicles currently traveling through it, and E represents a set of directed edges denoting traffic flow between segments. Each node in the graph corresponds to a road segment and encapsulates traffic condition data, allowing the graph G to model the global traffic status across the entire road network.

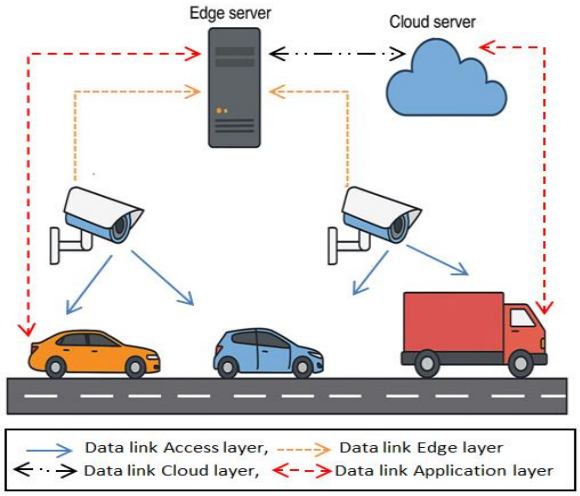


Figure 5.1: System environment

5.3.1 System Architecture and Communication Layers

As shown in Figure 5.1, the proposed system follows a modular and hierarchical design, organized into four main communication layers, each responsible for a specific function in data sensing, processing, and distribution:

Access Layer (Data Sensing Layer): This layer handles the interaction between roadside cameras and the road environment (i.e., vehicles). Cameras capture real-time visual data (images or video frames) of traffic conditions, including vehicle density, movement patterns, and road occupancy. This raw data forms the input for further analysis in upstream layers.

- **Edge Link Layer (Edge Data Link Layer):** This layer enables low-latency communication between cameras and nearby edge servers. Edge servers perform lightweight processing tasks such as vehicle detection, motion estimation, and local traffic evaluation. Communication typically occurs via high-speed wired links (e.g., Ethernet) or short-range wireless technologies (e.g., Wi-Fi, 5G).
- **Cloud Link Layer (Cloud Data Link Layer):** Responsible for transferring aggregated or preprocessed data from edge servers to the central cloud server. The cloud performs advanced analytics, such as machine learning-based predictions, traffic modeling, and long-term storage. Communication occurs over wide-area networks (WANs) or the internet and supports inter-regional coordination.
- **Application Layer (Data Interaction Layer):** This top layer connects the cloud/edge infrastructure with end users, including vehicles, drivers, and traffic management systems. It provides real-time feedback such as congestion alerts, rerouting suggestions, or traffic condition summaries. Applications may operate either on cloud platforms or at the edge, and users can access services through mobile apps or embedded vehicle systems via APIs or push notifications.

5.4 Multi-module Road Traffic Estimation approach (MRTE)

The proposed system consists of three modules. The first module characterizes the image and calculates the density of vehicles. The second module estimates the speed of vehicles and the third module provides the congestion level. Figure 5.2 shows the main components of the proposed system.

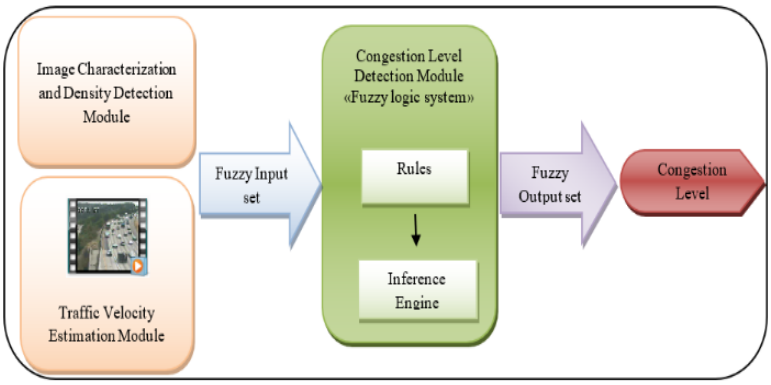


Figure 5.2 : MRTE framework

Let g be the image captured by the road camera and P be the set of key points p_i of this image g generated using the SIFT method using Eq (5.4). Here, $P_p = \{p_1, p_2, \ldots, p_l\}$, where k is the number of key points of image g and V_{pi} is the 128-dimensional descriptor vector associated with the key point P_i.

$$F_{sift}(g) = \begin{cases} p_i\ ,\ \ p_i \in P_p \\ V_{pi} = (a_{i1}, a_{i2}, \ldots., a_{i128}), a_{ij} \geq 0 \end{cases} \quad (5.1)$$

Therefore, image *g* can be represented by M_{SIFT} matrix. This matrix contains all the descriptor vectors of image *g*. Algorithm 5.1 shows the main steps of the M_{SIFT} construction process.

Algorithm 5.1 : Construction of the M_{SIFT} Matrix

Input: image g

Output: M_{SIFT} *matrix*

Start

1.Segment the image g to eliminate stationary objects (such as trees, etc.)

2.Apply the medium filter

3.Convert the RGB image to a grayscale image

4.Call the SIFT method

5.Construct the M_{SIFT} *matrix of image g as follows:*

$$M_{SIFT}(g) = \begin{pmatrix} a_{1,1} & \cdots & a_{1,128} \\ \vdots & \ddots & \vdots \\ a_{k,1} & \cdots & a_{k,128} \end{pmatrix}$$

End.

5.4.1 Vehicle Density measurement

Vehicle density refers to the number of vehicles presented in a road segment. In our approach, we do not focus on detecting individual vehicles or objects. Instead, we use the image representation approach represented by the M_{SIFT} matrix. The number of key points in the M_{SIFT} increases as the number of vehicles or other objects in the road segment increases. Therefore, as the number of vehicles increases, the size of the M_{SIFT} matrix increases accordingly. Consequently, we calculate the vehicle density *(V D)* according to the size of the M_{SIFT} and the values of its elements. Eq. (5.5) shows how *VD* is calculated.

$$VD = \frac{\mathrm{S_{MSIFT}}}{\mathrm{k} * 128} \quad (5.2)$$

Here, S_{MSIFT} is the sum of all M_{SIFT} matrix element values Eq. (5.6), *k* is the number of key points, and 128 is the size of the descriptor vector for each key point detected by the SIFT method.

$$\mathrm{S_{MSIFT}} = \sum_{i=1}^{k} \sum_{j=1}^{128} a_{i,j} \quad (5.3)$$

5.4.2. Traffic Velocity Estimation Module

In most traffic congestion detection systems, the speed of vehicles is considered as the primary input parameter. However, in many cases, vehicle speed measurements exhibit a significant error rate and require the deployment of hybrid systems that include cameras and sensors. Instead of focusing on calculating the speed of individual vehicles, our approach focuses on estimating the traffic speed. In this module, we calculate the speed of the entire traffic flow without relying on vehicle speed measurements. To achieve this, we periodically analyze the traffic road video recorded by roadside cameras. This task can be performed locally by the road camera. Traffic Velocity Estimation *(TVE)* is defined as the average rate of change of vehicle density *(VD)* for each image *g* in the recorded traffic road video. Figure 5.3 illustrates the *TVE* calculation process. Let *V* be a video recorded for a time period *D*, where *D* also represents the duration *V* of the video. *V* is then divided into *m* images or frames. Let $S_g = \{g_1, g_2, \ldots, g_m\}$ be the set of images that constitute the split video. In the literature, there are various ready-made software libraries available to split videos into frames [34,35]. For each image g_i of S_g, we calculate the intensity VD_{gi} using Eq. (5.5). Then, for each pair of consecutive *VD* values (VD_{gi} and VD_{gi+1}) for images g_i and g_{i+1}, respectively, we calculate the distance *Disti* using Eq. (5.7). *Disti* represents the absolute difference between VD_{gi} and VD_{gi+1}.

$$\mathrm{Dist_i} = |\mathrm{VD_i} - \mathrm{VD_{i+1}}|, 1 \le i \le m \tag{5.4}$$

Where *m* is the number of images of the split video. We define R_Dist_i as the ratio of distance $Dist_i$. R_Dist_i is calculated using Eq. (5.8).

$$\mathrm{R\_Dist_i} = \frac{\mathrm{Dist_i}}{\mathrm{Max\_Dist}} \tag{5.5}$$

Here *Max_Dist* is the maximum value of the distance set $(Dist_2, Dist_3, \ldots, Dist_m)$. Therefore, Traffic Velocity Estimation *(TVE)* is calculated as Eq. (5.9) shows.

$$\mathrm{TVE} = \frac{\sum_{i=2}^{m} \mathrm{R\_Dist_i}}{\mathrm{m} - 1} \tag{5.6}$$

Since consecutive images of the split video can report the same data, we apply Eq. (5.5) and (5.7) for a selected image list. To select this list, we apply translation or skip (image skip) of *(TVE)* images. If the current selected image is g_i, the next image is g_{i+k} . Therefore, the computation time of traffic speed is reduced, and the response time of the whole system can be improved.

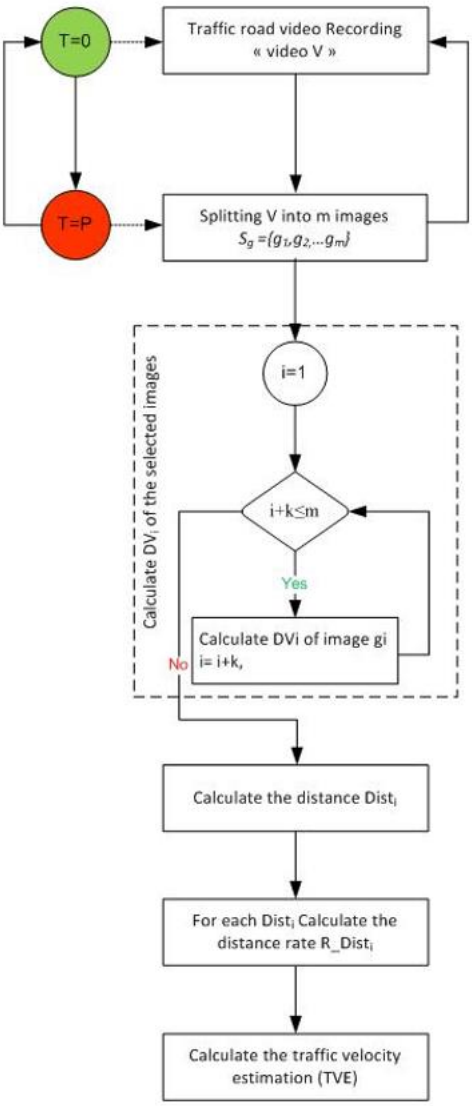


Figure 5.3: TVE estimation process

5.4.3. Congestion Level Detection Module

Traffic flow on roads can be divided into various categories according to the traffic congestion status. However, the change between traffic flow statuses is continuous in nature, which leads to uncertainty in defining the boundaries of congestion categories [167]. As a result, it becomes difficult to determine the boundaries between different traffic flow statuses. To address this uncertainty, especially when dealing with uncertain systems, fuzzy logic-based systems are widely used. We use fuzzy logic to determine the boundaries between traffic flow statuses. In our fuzzy logic system, we use two input variables: Vehicle Density *(VD)* and Traffic Velocity Estimation *(TVE)*. *DV* is classified into fuzzy sets such as "*low*", "*medium*", "*high*" and "*very high*", while *TVE* is associated with fuzzy sets representing "*very slow*", "*slow*", "*medium*" and "*fast*" traffic speeds. The output of our system determines the congestion level, which is called the congestion level output variable, which is classified into fuzzy sets such as "*very slow*",

"*slow*", "*medium*" and "*fast*" congestion. The principle of the fuzzy logic approach revolves around assigning membership functions to the values of each fuzzy set. Figure 5.4 and 5.5 show the membership functions used for input variables in our system.

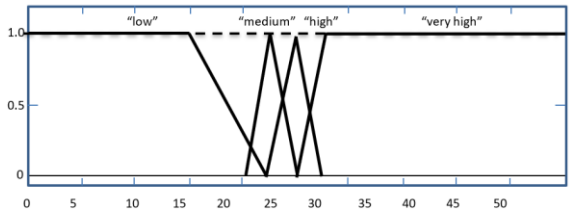


Figure 5.4: VD membership

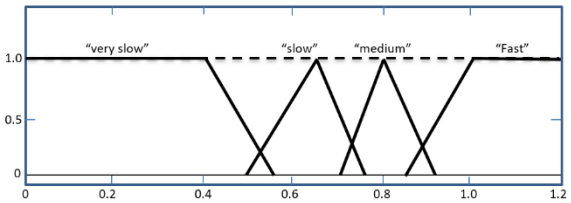


Figure 5.5: Traffic Velocity Estimation (TVE)

Different interval levels of traffic congestion are also described by the fuzzy set using four categories: "*free*"=[0,0.3],"*light*"=[0.2,0.6], "*medium*"=[0.5,0.9] and "*heavy*" =[0.8,1]. Figure 5.6 shows the membership function of congestion level.

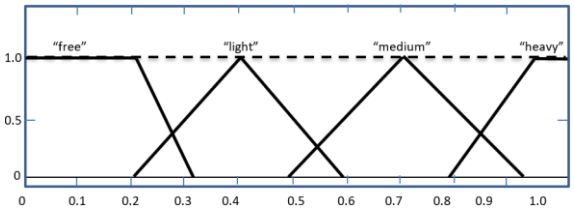


Figure 5.6: Congestion level membership

We define inference rules to complete the congestion level detection system.

1. If *TVE* is "*very slow*" and *VD* is "*low*", then congestion level is "*light*".
2. If *TVE* is "*very slow*" and *VD* is "*medium*", then congestion level is "*medium*".
3. If *TVE* is "*very slow*" and *VD* is "*high*", then congestion level is "*heavy*".
4. If *TVE* is "*very slow*" and *VD* is "*very high*", then congestion level is "*heavy*".
5. If *TVE* is "*slow*" and *VD* is "*low*", then congestion level is "*light*".
6. If *TVE* is "*slow*" and *VD* is "*medium*", then congestion level is "*light*".
7. If *TVE* is "*slow*" and *VD* is "*high*", then congestion level is "*medium*".
8. If *TVE* is "*slow*" and *VD* is "*very high*", then congestion level is "*heavy*".
9. If *TVE* is "*medium*" and *VD* is "*low*", then congestion level is "*free*".
10. If *TVE* is "*medium*" and *VD* is "*medium*", then congestion level is "*light*".
11. If *TVE* is "*medium*" and *VD* is "*high*", then congestion level is "*medium*".
12. If *TVE* is "*medium*" and *VD* is "*very high*", then congestion level is "*medium*".

13. If *TVE* is "*fast*" and *VD* is "*low*", then congestion level is "*free*".
14. If speed is "*fast*" and density is "*medium*", then congestion level is "*light*".
15. If speed is "*fast*" and density is "*high*", then congestion level is "*medium*".
16. If speed is "*fast*" and density is "*very high*", then congestion level is "*medium*".

Table 5.1 shows the rules that associate the input fuzzy sets (*VD* and *TVE*) with the output fuzzy sets for congestion level. Figure 5.7 shows the Surface Viewer rule for VD and TVE.

Table 5.1: Fuzzy logic rules

Congestion level		**VD**			
		Low	Medium	High	Very high
TVE	very slow	Light	Medium	Heavy	Heavy
	Slow	Light	Light	Medium	Heavy
	Medium	Free	Light	Medium	Medium
	Fast	Free	Free	Light	Medium

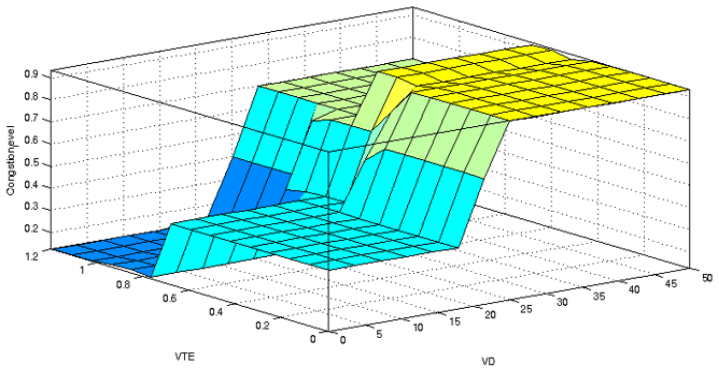


Figure 5.7: Rule surface viewer for VD and TVE (Congestion level)

5.5 Complexity Analysis of MRTE

In this section, we analyze the computational complexity of the Multi-module Road Traffic Estimation (MRTE) framework. The system comprises three sequential modules: (1) Image Characterization and Density Detection Module (ICDDM), (2) Traffic Velocity Estimation Module (TVE), and (3) Congestion Level Detection Module (CLD). We evaluate the complexity of each module based on its core operations.

5.5.1 Image Characterization and Density Detection (ICDDM)

This module utilizes the SIFT (Scale-Invariant Feature Transform) algorithm to extract keypoints and descriptors from grayscale images of road segments.

- **Input:** An image of size w×h.
- **Main operations:**
  - Gaussian filtering at multiple scales.
  - Keypoint detection across the scale-space.
  - Descriptor generation for each keypoint.

Let k be the number of keypoints per image. The complexity of SIFT is approximately O(k.l), where l=128 is the size of the descriptor vector per keypoint. In practice, since SIFT operates on dense feature maps at multiple scales, its computational complexity per image is O(n·logn), where n = w×h.

Complexity (per image):

$O(n \log n + l \cdot 128) \approx O(n \log n)$

5.5.2 Traffic Velocity Estimation (TVE)

TVE computes the rate of change in vehicle density across video frames.

- **Input:** m selected frames from a video.
- **Operations:**
  - VD computation using MSIFT matrix for each selected frame.
  - Compute pairwise differences between VD values.
  - Normalize and aggregate these values.

Let m be the number of sampled frames and k the number of skipped frames. TVE avoids per-frame processing by jumping k frames, reducing complexity to:

Complexity: $O((m/k) \cdot n \log n)$, assuming the same cost of SIFT per frame.

This results in near-linear complexity with respect to the number of selected frames and image size.

5.5.3 Congestion Level Detection (CLD)

This module applies a fuzzy inference system (FIS) with predefined fuzzy rules.

- **Input:** Two variables: VD and TVE.
- **Operations:**
  - Membership value computation (constant time).
  - Rule evaluation (16 rules).
  - Defuzzification.

As the number of fuzzy rules is constant and independent of the input size, the complexity is:

Complexity: $O(1)$

5.5.4 Total System Complexity

Assuming:

- I images per video (pre-sampling),
- F=I/k sampled frames used in TVE,

The total complexity per video is dominated by the two first modules:

$O(n \log n + F \cdot n \log n) = O(F \cdot n \log n)$

Where:

- n is the number of pixels per image,
- F is the number of frames used after skipping,
- The fuzzy logic module's contribution is negligible in asymptotic analysis.

The MRTE framework achieves near real-time performance due to:

- The ability to skip frames (k-frame sampling),
- Avoidance of object-level tracking or segmentation,
- A lightweight fuzzy inference engine.

The system thus offers a good balance between accuracy and computational cost, making it suitable for real-time traffic estimation tasks in resource-constrained or large-scale deployments.

5.6 Performance evaluation

In this section, we evaluate the performance of the proposed system using UCSD dataset [151]. We start by introducing metrics and then present and discuss the results.

5.6.1 Evaluation metrics

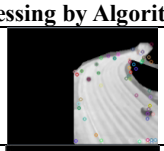
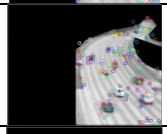
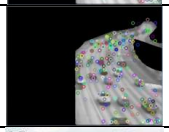
While evaluating the performance of our approach to classify traffic congestion into "*light*", "*moderate*" and "*heavy*" categories, we use the basic metrics such as precision, recall and F1 score. Mathematically, it is defined in the previous chapter, section 4.3.1 and accuracy is formally defined in chapter 3, section 3.3.1.

5.6.2 Discussion results

5.6.2.1 Vehicle density estimation

The purpose of this experiment is to demonstrate how to measure vehicle density on a road segment by calculating the *VD* using Eq. (5.5) of different images from the UCSD dataset. First, we extract four images representing different density levels ("*low*", "*medium*", "*high*", "*very high*") from the UCSD videos. Then, we apply Algorithm 1 to each type of images to calculate the S_{MSIFT} values. Table 5.2 presents the vehicle density *VD* value for each image type, while Figure 5.8 shows the variation of *VD* with respect to vehicle density.

Table 5.2: Vehicle density (VD)

	Images extracted from video UCSD dataset	Image view after processing by Algorithm 1	Vehicle density (VD) value
Low			21.5328
Medium			24.3798
High			26.2334
Very high			27.2266

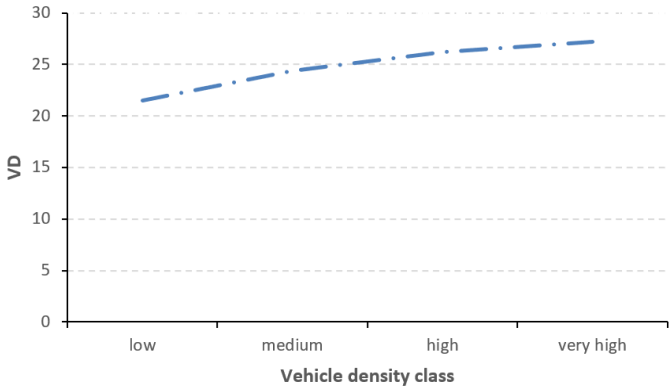


Figure 5.8: VD and vehicle density

The results in Figure 5.8 show that the *VD* values accurately reflect the vehicle density and the values increase proportionally to the number of vehicles present in the images.

5.6.2.2 Velocity estimation

In this experiment, we demonstrate the impact of vehicle speed on Traffic Velocity Estimation (*TVE*). We consider four different videos of different vehicle speed levels: "*low*", "*medium*", "*high*" and "*very high*". By applying our *TVE* method to each speed level, we aim to analyze how changes in vehicle speed affect the accuracy of traffic velocity estimation. The results will highlight the relationship between vehicle speed and *TVE* and provide insights into the performance of the method under changing traffic conditions. Table 5.3 shows the traffic speed estimate *TVE* for each video, while Figure 5.9 presents the results of the experiment. The results in Figure 5.9 clearly show that *TVE* increases with vehicle speed, confirming the expected relationship between vehicle speed and *TVE*. This finding highlights the effectiveness of the method in accurately estimating traffic speed under changing conditions.

Table 5.3: TVE for videos from the UCSD dataset

Vehicle speed in the video	TVE
Very slow	0.405
Slow	0.656
Medium	0.814
Fast	1.118

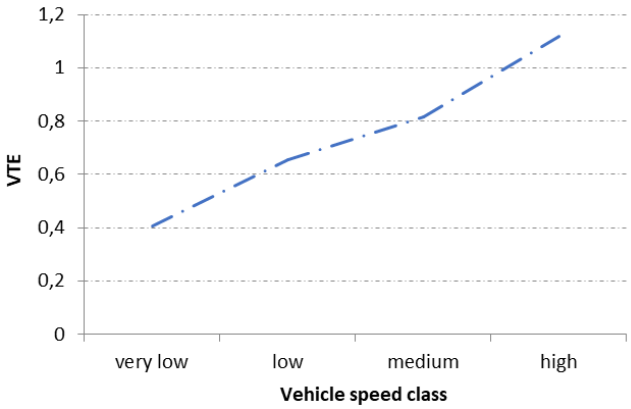


Figure 5.9: TVE and vehicle speed

5.6.2.3 Traffic classification capability

In this experiment, we evaluate the capability of MRTE in detecting traffic congestion classes. The system was executed using images representing different traffic conditions, light, medium, and heavy, and its performance was measured using standard classification metrics: accuracy, precision, recall, and F1-score.

The results, presented in Table 5.4, highlight the strong performance of the framework in classifying traffic congestion levels. For the “light” congestion class, the system achieves perfect scores across all metrics (100% accuracy, precision, recall, and F1-score), indicating error-free classification. This demonstrates that the model is highly effective in distinguishing light congestion from other classes.

Table 5.4: Performance of the MRTE framework

Metric / Class	Light	Medium	Heavy	Average
Accuracy (%)	100	99.2	99.2	99.47
Precision (%)	100	100	95.7	98.57
Recall (%)	100	95.6	100	98.53
F1-score (%)	100	97.7	97.8	98.50

For the “medium” congestion class, the MRTE maintains high performance, with accuracy between 95.6% and 99.2%. While the precision remains excellent, the recall is slightly lower. This suggests that the system is highly precise in labeling “medium” congestion (rarely

misclassifying other classes as medium) but may occasionally fail to detect some medium congestion instances, resulting in a few false negatives. Nevertheless, the F1-score of 97.7% reflects a strong balance between precision and recall.

The heavy congestion class also shows high performance with 99.2% accuracy, 95.7% precision, 100% recall, and an F1-score of 97.8%. This reflects the framework's ability to correctly identify all "heavy" congestion cases (perfect recall), while it is slightly less precise due to a few "medium" cases being incorrectly classified as heavy. The high F1-score suggests that the model effectively manages this fine trade-off between precision and recall. Overall, the average metrics: 99.47% accuracy, 98.57% precision, 98.53% recall and 98.5% F1-score demonstrate the model's consistent and robust performance across all congestion levels. The nearly perfect accuracy and robust "average" scores suggest that the model can reliably classify traffic congestion with minimal error. The slight differences in precision and recall for "medium" and "heavy" congestion suggest that while the framework is highly effective, there is a slight difficulty in distinguishing between these two classes, likely due to their closer similarity compared to light congestion. However, the overall performance is exemplary, making the framework a reliable tool for real-world traffic congestion classification. The confusion matrix shown in Figure 5.10 shows that the MRTE framework performs exceptionally well in predicting "light" traffic with 165 correct predictions and only 1 misclassification.

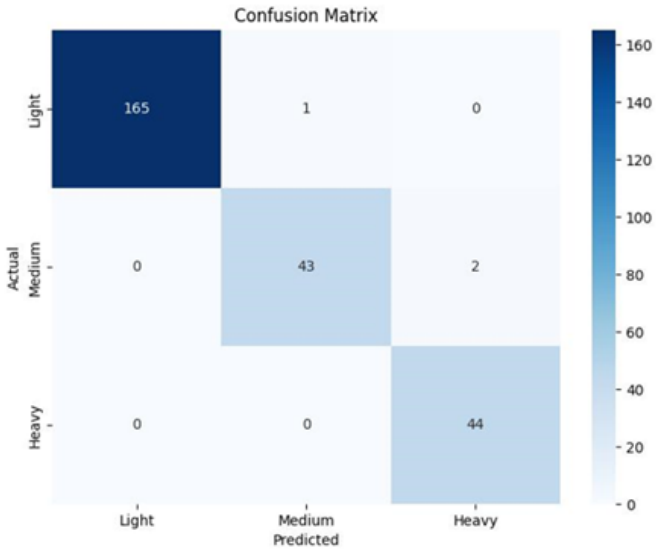


Figure 5.10: Confusion matrix of the results of the MRTE framework

For "medium" traffic, the model correctly classified 43 instances but made 2 errors by classifying "medium" traffic as "heavy". The framework is also quite accurate in predicting "heavy" traffic with 44 correct predictions and no misclassifications as "light" or "medium".

These results show that the model is quite effective in distinguishing “light” traffic, with minor difficulties in distinguishing between “medium” and “heavy” traffic conditions.

5.6.2.4 Execution time

Our system consists of three modules and the overall execution time depends on the time taken by each module. To estimate this time, we consider the following: Assume that T_{m1}, T_{m2} and T_{m3} represent the execution times for the Image Characterization and Density Detection Module (ICDDM), Traffic Velocity Estimation (*TVE*) module and Congestion Level Detection (*CLD*) module, respectively. The UCSD dataset contains 254 videos, each 5 seconds long and each video is divided into 50 images. These images are very similar to each other and the vehicle density (*DV*) values are closely related. The execution time for the second module *TVE* is estimated by processing only one frame per second, resulting in 5 frames per video. The total system execution time, Ts is calculated using Eq. (5.14).

$$T_s = T_{m1} + T_{m2} + T_{m3} \tag{5.7}$$

To calculate the total system execution time, we select three types of videos (“light”, “medium”, “heavy”) from the UCSD dataset according to their congestion levels. The results of these experiments are presented in Table 5.5.

Table 5.5 : System execution time

Video type	T_{m1}(s)	T_{m2} (s)	T_{m3} (s)	T_s (s)
light	0.12	1.92	0.01	2.05
medium	0.14	2.24	0.01	2.39
heavy	0.16	2.56	0.01	2.73

According to the results presented in Table 5.5, the execution time for each module: T_{m1}, T_{m2}, and T_{m3} is within an acceptable range and meets the expectations of our system. The total system execution time T_s calculated using Eq. (5.14) confirms that our approach handles the processing tasks efficiently. Despite the complexity of processing video data and calculating vehicle density and traffic speed, the system responds quickly, making it suitable for real-time traffic monitoring and analysis. The performance of our system is well aligned with its intended purpose, providing reliable and timely detection of traffic conditions.

5.6.2.5 Traffic Velocity Estimation (TVE) and image jump (k)

Each video in the UCSD dataset is 5 seconds long and 50 frames, i.e. 10 frames per second. This part of the experiment investigates the effect of frame jump (*k*) on Traffic Velocity Estimation (*TVE*). Parameter k is defined in the second module of our system. We selected four videos with different speeds and varied the *k* value from *0* to *5*, which corresponds to *0* to *5* frames skipping during processing. We calculated the *TVE* of each video for each *k* value. In Fig. 11 we observe that the *TVE* value is not affected up to $k = 4$ for all video types and values

below k. Figure 5.11 shows the results found with the curves plotted according to the video type. The “High” category shows a significant upward trend indicating a strong positive correlation with increasing *k*. The “Medium” category shows moderate fluctuations peaking around *k*=3, but generally remains relatively stable. The “Low” category shows minimal changes around 0.5, while the “Very low” category shows a slight, steady increase, but remains relatively stable.

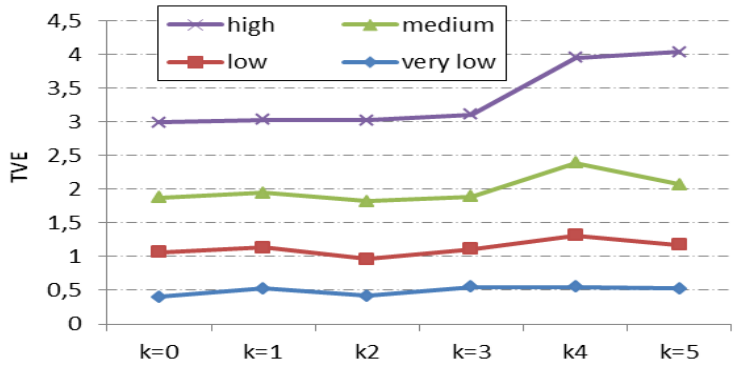


Figure 5.11: TVE and image hop (k)

This result shows that the effect of increasing *k* is most pronounced in the “High” category, with less variability observed in the other categories. Therefore, we conclude that *k*=3 allows skipping 3 frames and estimating the vehicle speed in the video sequence without degrading the system performance.

5.7 Conclusion

This chapter introduces the Multi-Module Road Traffic Estimation System (MRTE) for real-time traffic congestion detection using the SIFT algorithm. By integrating vehicle density and Velocity Estimation with fuzzy logic, MRTE offers a robust approach to traffic management, demonstrating high accuracy across various conditions. MRTE employs an MSIFT matrix to characterize vehicle density and estimate speed, integrating these metrics into a fuzzy inference system (FIS) to assess congestion levels. Experimental results demonstrate that MRTE achieves high accuracy and efficiency for classifying traffic congestion levels. MRTE’s implementation in smart cities has significant potential to optimize traffic management, helping authorities mitigate delays and reduce CO2 emissions by rerouting drivers during congestion.

Chapter 6 : Conclusion and future work

6.1 Conclusion

This thesis developed and tested advanced artificial intelligence models to improve intelligent surveillance for smart cities. The work focused on two key areas of Intelligent Transportation Systems (ITS): parking management and traffic congestion control. The proposed solutions were designed to be accurate, efficient, and practical for real-world use.

To address these challenges, different artificial intelligence models and frameworks were created and tested. Each contribution targeted a specific problem while also working towards a complete vision for Intelligent Transportation Systems.

A custom Convolutional Neural Network (CNN) was designed using dilated convolutions. This method improved accuracy while keeping the model lightweight (fewer computations and parameters). Because of its efficiency, this model can run on small devices such as smart cameras or edge systems placed directly in parking areas. This makes it highly practical for real-world deployment.

Besides the custom CNN, transfer learning was also explored using the Inception V3 model. By applying data augmentation (creating variations of training images), the model adapted well to different parking scenarios. This showed that pre-trained models can be reused and adjusted for specific urban problems, saving time and resources while achieving strong performance.

A novel pipeline was developed combining YOLOv8s for object detection and a lightweight Vision Transformer (UniViT) for classification. YOLOv8s transformed raw video frames into simplified vehicle bounding boxes, reducing the amount of information the transformer needed to process. UniViT, with its efficient attention mechanism, achieved 99.89% accuracy on the UCSD dataset, even in different weather conditions. This proved that transformer-based models can bring major improvements to traffic analysis tasks.

While deep learning is powerful, it is often seen as a “black box.” To address this, the Multi-Module Road Traffic Estimation (MRTE) framework was developed. It uses lightweight SIFT-based image processing and a fuzzy logic inference engine to estimate congestion levels based on vehicle density and speed. MRTE does not require large annotated datasets or expensive training. More importantly, its results are transparent and easy to understand, making it useful for traffic authorities who need clear and interpretable information for decision-making.

In conclusion, this thesis makes three main contributions:

- Lightweight CNN models (custom and transfer learning-based) for accurate and real-time parking detection.
- High-accuracy traffic congestion classification using UniViT with an efficient image mapping approach.
- Interpretable and efficient traffic analysis through the MRTE fuzzy logic framework, offering a transparent alternative to data-intensive deep learning.

6.2 Future Work

Although this research achieved significant progress, several opportunities remain for further development:

- A unified ITS vision: In this thesis, the parking detection models, congestion classification model, and MRTE framework were developed and validated separately. A key future direction is to integrate these independent modules into a single, unified ITS platform. Such a system would allow the models to complement each other, helping to reduce both parking search time and traffic congestion in a coordinated way. This integrated framework could form the basis of a comprehensive citywide dashboard for urban mobility management.
- Improving generalization: Future models should be tested in more diverse and unseen environments, including under extreme weather conditions (e.g., heavy rain, fog, snow), to ensure reliability across different cities.
- Reducing latency: Further optimization is needed to achieve ultra-real-time performance, which is crucial for critical tasks like adaptive traffic signal control or emergency vehicle routing.
- Temporal data and forecasting: Adding temporal information would enable short-term traffic predictions, giving city authorities and drivers the ability to anticipate and respond proactively to changing traffic conditions.
- Pilot studies in real smart cities: Testing these models in real-world environments is essential to evaluate their effectiveness under real traffic conditions, hardware limitations, and dynamic human behavior.

Bibliography

[1] United Nations. (2022). "World Urbanization Prospects: The 2022 Revision." United Nations Department of Economic and Social Affairs. https://www.un.org/uk/desa/68-world-population-projected-live-urban-areas-2050-says-un , Last accessed : August 7, 2025.

[2] INRIX. (2025). 2025 Global Traffic Scorecard. INRIX, Inc. https://inrix.com/scorecard/ , Last accessed : August 14, 2025.

[3] Shoup, Donald. "Pricing curb parking." Transportation Research Part A: Policy and Practice 154 (2021): 399-412.

[4] Xie, L., Hong, R., Zhang, B., & Tian, Q. "Image classification and retrieval are one". In Proceedings of the 5th ACM on International Conference on Multimedia Retrieval, 2015. p. 3-10.

[5] Abraham, A., Prasad, S., Kashyap Pargi, M., Alhammadi, A., & Vyas, P. "Computer Vision in Smart Parking Solution Systems: Enhancing Urban Mobility." Internet of Vehicles and Computer Vision Solutions for Smart City Transformations. Cham: Springer Nature Switzerland, 2025. 217-242.

[6] Laña, I., Sanchez-Medina, J. J., Vlahogianni, E. I., & Del Ser, J. "From data to actions in intelligent transportation systems: A prescription of functional requirements for model actionability." Sensors 21.4 (2021): 1121.

[7] Khan, AB Feroz, and P. Ivan. "Integrating machine learning and deep learning in smart cities for enhanced traffic congestion management: An empirical review." J. Urban Dev. Manag 2.4 (2023): 211-221.

[8] Zhao, X., Wang, L., Zhang, Y., Han, X., Deveci, M., & Parmar, M. "A review of convolutional neural networks in computer vision." Artificial Intelligence Review 57.4 (2024): 99.

[9] Alan M. Turing. "Computing Machinery and Intelligence". Mind, Oxford, UK, 1950.

[10] Mohamed, Serrhini. "Maturity and Future of Artificial Intelligence." New Journal for Science Vulgarization 2.1 (2022): 52-70.

[11] Nikoleta Manakitsa, George S Maraslidis, Lazaros Moysis, and George F Fragulis. "A review of machine learning and deep learning for object detection, semantic segmentation, and human action recognition in machine and robotic vision". Technologies, 12(2):15, 2024.

[12] José Manuel Amigo. "Data mining, machine learning, deep learning, chemometrics: Definitions, common points and trends (spoiler alert: Validate your models!)". Brazilian Journal of Analytical Chemistry, 8(32):45–61, 2021.

[13] Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. "Activation functions: Comparison of trends in practice and research for deep learning" (2018). arXiv preprint arXiv:1811.03378, 106, 1811.

[14] Hieu Tran. "A survey of machine learning and data mining techniques used in multimedia system". Dept. Comput. Sci., Univ. Texas Dallas Richardson, Richardson, TX, USA, Tech. Rep, 2019.

[15] Batta Mahesh. "Machine learning algorithms-a review". International Journal of Science and Research (IJSR).[Internet], 9(1):381–386, 2020.

[16] Jafar Alzubi, Anand Nayyar, and Akshi Kumar. "Machine learning from theory to algorithms: an overview". In Journal of physics: conference series, volume 1142, page 012012. IOP Publishing, 2018.

[17] Manuel Méndez, Mercedes G Merayo, and Manuel Núñez. "Machine learning algorithms to forecast air quality: a survey". Artificial Intelligence Review, 56(9):10031–10066, 2023.

[18] Lin, Jyh-Woei. "Artificial neural network related to biological neuron network: a review." Advanced Studies in Medical Sciences 5.1 (2017): 55-62.

[19] Henrique F de Arruda, Alexandre Benatti, César Henrique Comin, and Luciano da F Costa. "Learning deep learning". Revista Brasileira de Ensino de Física, 44:e20220101, 2022.

[20] Leiyu Chen, Shaobo Li, Qiang Bai, Jing Yang, Sanlong Jiang, and Yanming Miao. "Review of image classification algorithms based on convolutional neural networks". Remote Sensing, 13(22):4712, 2021.

[21] Yirong Xie andothers. "Automated design of CNN architecture based on efficient evolutionary search". In Neurocomputing: 491 (2022), pages 160–171.

[22] Mohammed, F. A., Tune, K. K., Assefa, B. G., Jett, M., & Muhie, S. "Medical Image Classifications Using Convolutional Neural Networks: A Survey of Current Methods and Statistical Modeling of the Literature". inMachine Learning and Knowledge Extraction: 6.1 (2024), pages 699–735.

[23] Marina Adriana Mercioni and Stefan Holban. "The most used activation functions: Classic versus current". In 2020 International Conference on Development and Application Systems (DAS), pages 141–145. IEEE, 2020.

[24] JiUn Hong, Saad Arslan, TaeGeon Lee, and HyungWon Kim. "Design of power-efficient training accelerator for convolution neural networks". Electronics, 10(7):787, 2021.

[25] Aya Hassan, Moatamed Refaat, and Ashraf M Hemeida. "Image classification based deep learning: A review". Aswan University Journal of Sciences and Technology, 2(1):11–35, 2022.

[26] S. Chatterjee, D. Hazra, Y.-C. Byun, and Y.-W. Kim. "Enhancement of image classification using transfer learning and gan-based synthetic data augmentation." Mathematics, vol. 10, no. 9, p. 1541, 2022.

[27] Priyadarshan Patil. "Applications of deep learning in traffic management: A review". International Journal of Business Intelligence and Big Data Analytics, 5(1):16–23, 2022.

[28] Gudala Lavanya and Sagar Dhanraj Pande. "Enhancing real-time object detection with yolo algorithm". EAI Endorsed Transactions on Internet of Things, 10, 2024.

[29] Haitong Lou, Xuehu Duan, Junmei Guo, Haiying Liu, Jason Gu, Lingyun Bi, and Haonan Chen. "Dc-yolov8: small-size object detection algorithm based on camera sensor". Electronics, 12(10):2323, 2023.

[30] Bhagya S Hadagalimath, Ayesha Siddiqui, and Suvarna Nandyal. "Classification and counting of vehicle using image processing techniques". 2022.

[31] Muhammad Hussain. "Yolo-v1 to yolo-v8, the rise of yolo and its complementary nature toward digital manufacturing and industrial defect detection". Machines, 11(7):677, 2023.

[32] Youshan Zhang. "Stall number detection of cow teats key frames". arXiv preprint arXiv:2303.10444, 2023.

[33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". Advances in neural information processing systems, 30, 2017.

[34] Utpal Barman, Parismita Sarma, Mirzanur Rahman, Vaskar Deka,Swati Lahkar, Vaishali Sharma, and Manob Jyoti Saikia. "Vit-smartagri:Vision transformer and smartphone-based plant disease detection for smart agriculture". Agronomy, 14(2):327, 2024.

[35] Dosovitskiy, Alexey. "An image is worth 16x16 words: Transformers for image recognition at scale." arXiv preprint arXiv:2010.11929 (2020).

[36] Lowe, David G. "Object recognition from local scale-invariant features." Proceedings of the seventh IEEE international conference on computer vision. Vol. 2. Ieee, 1999.

[37] TANG, Vanessa WS, ZHENG, Yuan, et CAO, Jiannong. "An intelligent car park management system based on wireless sensor networks". In : 2006 First International Symposium on Pervasive Computing and Applications. IEEE, 2006. p. 65-70.

[38] Srikanth, S. V., Pramod, P. J., Dileep, K. P., Tapas, S., Patil, M. U., & Sarat, C. B. N. "Design and implementation of a prototype smart parking (spark) system using wireless sensor networks". In : 2009 International Conference on Advanced Information Networking and Applications Workshops. IEEE, 2009. p. 401-406.

[39] CHINRUNGRUENG, Jatuporn, SUNANTACH AIKUL, Udomporn, et TRIAMLUMLERD, Satien. "Smart parking: An application of optical wireless sensor network". In : 2007 International Symposium on Applications and the Internet Workshops. IEEE, 2007. p. 66-66.

[40] LEE, Sangwon, YOON, Dukhee, et GHOSH, Amitabha. "Intelligent parking lot application using wireless sensor networks". In : CTS. 2008. p. 48-57.

[41] TANG, Vanessa WS, ZHENG, Yuan, et CAO, Jiannong. "An intelligent car park management system based on wireless sensor networks". In : 2006 First International Symposium on Pervasive Computing and Applications. IEEE, 2006. p. 65-70.

[42] Wolff, J., Heuer, T., Gao, H., Weinmann, M., Voit, S., & Hartmann, U. "Parking monitor system based on magnetic field senso". In : 2006 IEEE Intelligent Transportation Systems Conference. IEEE, 2006. p. 1275-1279.

[43] CHUNHE, Yu et JILIN, Liu. "A type of sensor to detect occupancy of vehicle berth in carpark". In : Proceedings 7th International Conference on Signal Processing, 2004. Proceedings. ICSP'04. 2004. IEEE, 2004. p. 2708-2711.

[44] KUMAR, Rakesh, CHILAMKURTI, Naveen K., et SOH, Ben. "A comparative study of different sensors for smart car park management". In : the 2007 International Conference on Intelligent Pervasive Computing (IPC 2007). IEEE, 2007. p. 499-502.

[45] Ojansivu, Ville, and Janne Heikkilä. "Blur insensitive texture classification using local phase quantization." International conference on image and signal processing. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[46] Ojala, Timo, and Matti Pietikäinen. "Unsupervised texture segmentation using feature distributions." Pattern recognition 32.3 (1999): 477-486.

[47] Dalal, Navneet, and Bill Triggs. "Histograms of oriented gradients for human detection." 2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05). Vol. 1. Ieee, 2005.

[48] Lan, Rushi, Yicong Zhou, and Yuan Yan Tang. "Quaternionic local ranking binary pattern: a local descriptor of color images." IEEE Transactions on Image Processing 25.2 (2015): 566-579.

[49] Almeida, P., Oliveira, L. S., Silva, E., Britto, A., & Koerich, A. "Parking space detection using textural descriptors." 2013 IEEE International Conference on Systems, Man, and Cybernetics. IEEE, 2013.

[50] De Almeida, P. R., Oliveira, L. S., Britto Jr, A. S., Silva Jr, E. J., & Koerich, A. L. "PKLot–A robust dataset for parking lot classification." Expert Systems with Applications 42.11 (2015): 4937-4949.

[51] Almeida, P. R., Oliveira, L. S., Britto Jr, A. S., & Sabourin, R. "Adapting dynamic classifier selection for concept drift." Expert Systems with Applications 104 (2018): 67-85.

[52] de Almeida, P. R. L., Oliveira, L. S., de Souza Britto, A., & Barddal, J. P. "Naïve approaches to deal with concept drifts." 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC). IEEE, 2020.

[53] Suwignyo, Markus Antoni, Iwan Setyawan, and Banu Wirawan Yohanes. "Parking space detection using quaternionic local ranking binary pattern." 2018 International Seminar on Application for Technology of Information and Communication. IEEE, 2018.

[54] Hammoudi, K., Cabani, A., Melkemi, M., Benhabiles, H., & Windal, F. "Towards a model of car parking assistance system using camera networks: Slot analysis and communication management." 2018 IEEE 20th International Conference on High Performance Computing and Communications; IEEE 16th International Conference on Smart City; IEEE 4th International Conference on Data Science and Systems (HPCC/SmartCity/DSS). IEEE, 2018.

[55] Hammoudi, K., Melkemi, M., Dornaika, F., Benhabiles, H., Windal, F., & Taoufik, O. "A comparative study of 2 resolution-level LBP descriptors and compact versions for visual analysis." International Conference on Multimedia and Ubiquitous Engineering. Singapore: Springer Singapore, 2018.

[56] Hammoudi, K., Melkemi, M., Dornaika, F., Phan, T. D. A., & Taoufik, O. "Computing multi-purpose image-based descriptors for object detection: powerfulness of LBP and its variants." Third International Congress on Information and Communication Technology: ICICT 2018, London. Singapore: Springer Singapore, 2018.

[57] Hammoudi, K., Abu Taha, M., Benhabiles, H., Melkemi, M., Windal, F., El Assad, S., & Queudet, A. "Image-based ciphering of video streams and object recognition for urban and vehicular surveillance services." Fourth International Congress on Information and Communication Technology: ICICT 2019, London, Volume 2. Singapore: Springer Singapore, 2020.

[58] Amato, G., Carrara, F., Falchi, F., Gennaro, C., Meghini, C., & Vairo, C. "Deep learning for decentralized parking lot occupancy detection". Expert Systems with Applications, 72, 327–334. (2017).

[59] Dizon, C. C., Magpayo, L. C., Uy, A. C., & Tiglao, N. M. C. "Development of an open-space visual smart parking system." 2017 International Conference on Advanced Computing and Applications (ACOMP). IEEE, 2017.

[60] Thike, Linn Linn, and Thin Lai Lai Thein. "Parking space detection using complemented-ULBP background subtraction." 2019 IEEE 8th Global Conference on Consumer Electronics (GCCE). IEEE, 2019.

[61] Amato, G., Carrara, F., Falchi, F., Gennaro, C., & Vairo, C. "Car parking occupancy detection using smart camera networks and deep learning". In 2016 IEEE symposium on computers and communication (ISCC), Vol. 2016-August (pp. 1212–1217) (2016).. Institute of Electrical and Electronics Engineers Inc..

[62] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. "Imagenet: A large-scale hierarchical image database." 2009 IEEE conference on computer vision and pattern recognition. Ieee, 2009.

[63] Dornaika, F., Hammoudi, K., Melkemi, M., & Phan, T. D. A. "An efficient pyramid multi-level image descriptor: application to image-based parking lot monitoring." Signal, Image and Video Processing 13.8 (2019): 1611-1617.

[64] Irfan, A. P., N. Nur, and F. Wajidi. "Parking slot detection using glcm and similarity measure." IOP Conference Series: Materials Science and Engineering. Vol. 875. No. 1. IOP Publishing, 2020.

[65] Mago, Neeru, and Satish Kumar. "Role of computers in material science and design of classification model to search for the vacancy in outdoor parking lots." Materials Today: Proceedings 28 (2020): 1376-1381.

[66] Ahrnbom, Martin, Kalle Astrom, and Mikael Nilsson. "Fast classification of empty and occupied parking spaces using integral channel features." Proceedings of the IEEE conference on computer vision and pattern recognition workshops. 2016.

[67] Baroffio, L., Bondi, L., Cesana, M., Redondi, A. E., & Tagliasacchi, M. "A visual sensor network for parking lot occupancy detection in smart cities". In : 2015 IEEE 2nd World Forum on Internet of Things (WFIoT). IEEE, 2015. p. 745-750

[68] Bondi, L., Baroffio, L., Cesana, M., Redondi, A., & Tagliasacchi, M. "EZ-VSN: an open-source and flexible framework for visual sensor networks." IEEE Internet of Things Journal 3.5 (2015): 767-778.

[69] Vítek, Stanislav, and Petr Melničuk. "A distributed wireless camera system for the management of parking spaces." Sensors 18.1 (2017): 69.

[70] Hadi, Raad Ahmed, and Loay Edwar George. "Vision-based parking lots management system using an efficient adaptive weather analytic technique." 2019 12th International Conference on Developments in eSystems Engineering (DeSE). IEEE, 2019.

[71] Ding, Lijun, and Ardeshir Goshtasby. "On the Canny edge detector." Pattern recognition 34.3 (2001): 721-725.

[72] Raj, S. U., Manikanta, M. V., Harsitha, P. S. S., & Leo, M. J. "Vacant parking lot detection system using random forest classification." 2019 3rd International Conference on Computing Methodologies and Communication (ICCMC). IEEE, 2019.

[73] Varghese, Arun, and G. Sreelekha. "An efficient algorithm for detection of vacant spaces in delimited and non-delimited parking lots." IEEE Transactions on Intelligent Transportation Systems 21.10 (2019): 4052-4062.

[74] Goez Mora, Jhon Edison, Juan Camilo Londoño Lopera, and Diego Alberto Patiño Cortes. "Automatic visual classification of parking lot spaces: A comparison between bof and cnn approaches." Workshop on Engineering Applications. Cham: Springer International Publishing, 2018.

[75] Bay, H., Ess, A., Tuytelaars, T., & Van Gool, L. "Speeded-up robust features (SURF)." Computer vision and image understanding 110.3 (2008): 346-359.

[76] Pouyanfar, S., Sadiq, S., Yan, Y., Tian, H., Tao, Y., Reyes, M. P., & Iyengar, S. S. "A Survey on Deep Learning: Algorithms, Techniques, and Applications", ACM Comput. Surv., vol. 51, no. 5, Article 92, 2018.

[77] Dargan, S., Kumar, M., Ayyagari, M. R., & Kumar, G. "A Survey of Deep Learning and Its Applications: A New Paradigm to Machine Learning", Archives of Computational Methods in Engineering, vol. 27, no. 4, pp. 1071-1092, 2020.

[78] Ren, Z., Lai, J., Wu, Z., & Xie, S. "Deep neural networks-based real-time optimal navigation for an automatic guided vehicle with static and dynamic obstacles." Neurocomputing 443 (2021): 329-344.

[79] Bouwmans, T., Javed, S., Sultana, M., & Jung, S. K. "Deep neural network concepts for background subtraction: A systematic review and comparative evaluation." Neural Networks 117 (2019): 8-66.

[80] Kasera, Rohit Kumar, and Tapodhir Acharjee. "Parking slot occupancy prediction using LSTM." Innovations in Systems and Software Engineering 21.1 (2025): 65-77.

[81] Kasera, Rohit Kumar, and Tapodhir Acharjee. "A smart indoor parking system." SN Computer Science 3.1 (2022): 9.

[82] KRIZHEVSKY, Alex, SUTSKEVER, Ilya, et HINTON, Geoffrey E. "Imagenet classification with deep convolutional neural networks". In : Advances in neural information processing systems. 2012. p. 1097-1105.

[83] Di Mauro, D., Battiato, S., Patanè, G., Leotta, M., Maio, D., Farinella, G.M., "Learning Approaches for Parking Lots Classification", In: Blanc-Talon, J., Distante, C., Philips, W., Popescu, D., Scheunders, P. (eds) Advanced Concepts for Intelligent Vision Systems. ACIVS 2016. Lecture Notes in Computer Science(), Springer, Cham, vol 10016, 2016.

[84] Amato, G., Bolettieri, P., Moroni, D., Carrara, F., Ciampi, L., Pieri, G., ... & Vairo, C. "A wireless smart camera network for parking monitoring." 2018 IEEE globecom workshops (GC wkshps). IEEE, 2018.

[85] Rahman, S., Ramli, M., Arnia, F., Sembiring, A., & Muharar, R. "Convolutional neural network customization for parking occupancy detection." 2020 International Conference on Electrical Engineering and Informatics (ICELTICs). IEEE, 2020.

[86] Nguyen, T., Tran, T., Mai, T., Le, H., Le, C., Pham, D., & Phung, K. H. "An adaptive vision-based outdoor car parking lot monitoring system." 2020 IEEE Eighth International Conference on Communications and Electronics (ICCE). IEEE, 2021.

[87] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. "Mobilenets: Efficient convolutional neural networks for mobile vision applications." arXiv preprint arXiv:1704.04861 (2017).

[88] Bura, H., Lin, N., Kumar, N., Malekar, S., Nagaraj, S., & Liu, K. "An edge based smart parking solution using camera networks and deep learning." 2018 IEEE international conference on cognitive computing (ICCC). IEEE, 2018.

[89] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. "Gradient-based learning applied to document recognition." Proceedings of the IEEE 86.11 (2002): 2278-2324

[90] Nyambal, Julien, and Richard Klein. "Automated parking space detection using convolutional neural networks." 2017 Pattern Recognition Association of South Africa and Robotics and Mechatronics (PRASA-RobMech). IEEE, 2017.

[91] Polprasert, C., Sruayiam, C., Pisawongprakan, P., & Teravetchakarn, S. "A camera-based smart parking system employing low-complexity deep learning for outdoor environments." 2019 17th International Conference on ICT and Knowledge Engineering (ICT&KE). IEEE, 2019.

[92] Kolhar, Manjur, and A. Alameen, “Multi criteria decision making system for parking system,” Computer Systems Science and Engineering 36, no. 1 (2021): 101-116.

[93] Redmon, Joseph, and A. Farhadi, “Yolov3: An incremental improvement,” arXiv preprint arXiv:1804.02767 (2018).

[94] Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. "How transferable are features in deep neural networks?." Advances in neural information processing systems 27 (2014).

[95] Valipour, S., Siam, M., Stroulia, E., & Jagersand, M. "Parking-stall vacancy indicator system, based on deep convolutional neural networks." 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT). IEEE, 2016.

[96] Zhang, Wenjin, Jason Yan, and Cui Yu. "Smart parking system based on convolutional neural network models." 2019 6th International Conference on Information Science and Control Engineering (ICISCE). IEEE Computer Society, 2019.

[97] Simonyan, Karen, and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition." arXiv preprint arXiv:1409.1556 (2014).

[98] Dhuri, Vighnesh, A. Khan, Y. Kamtekar, D. Patel, and I. Jaiswal, "Realtime parking lot occupancy detection system with vgg16 deep neural network using decentralized processing for public, private parking facilities, " In 2021 International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), pp. 1-8. IEEE, 2021.

[99] Acharya, Debaditya, Weilin Yan, and Kourosh Khoshelham. "Real-time image-based parking occupancy detection using deep learning." Research@ Locate 4 (2018): 33-40.

[100] Jensen, T. H., Schmidt, H. T., Bodin, N. D., Nasrollahi, K., & Moeslund, T. B. (2017). "Parking space occupancy verification-improving robustness using a convolutional neural network." International Conference on Computer Vision Theory and Applications. Vol. 6. SciTePress, 2017.

[101] Nurullayev, Sherzod, and Sang-Woong Lee. "Generalized parking occupancy analysis based on dilated convolutional neural network." Sensors 19.2 (2019): 277.

[102] Ding, Xiangwu, and Ruidi Yang. "Vehicle and parking space detection based on improved yolo network model." Journal of physics: conference series. Vol. 1325. No. 1. IOP Publishing, 2019.

[103] Redmon, Joseph, and Ali Farhadi. "Yolov3: An incremental improvement." arXiv preprint arXiv:1804.02767 (2018).

[104] Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. "The pascal visual object classes (voc) challenge." International journal of computer vision 88.2 (2010): 303-338.

[105] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. "Microsoft coco: Common objects in context." European conference on computer vision. Cham: Springer International Publishing, 2014.

[106] He, K., Zhang, X., Ren, S., & Sun, J. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[107] Gregor, Michal, Rastislav Pirník, and Dušan Nemec. "Transfer learning for classification of parking spots using residual networks." Transportation Research Procedia 40 (2019): 1327-1334.

[108] Ebren, Ayşe Betül, and Bülent Bolat. "Determining the occupancy of Vehicle Parking Areas by deep learning." 2020 International Conference on Electrical, Communication, and Computer Engineering (ICECCE). IEEE, 2020.

[109] Thakur, N., Bhattacharjee, E., Jain, R., Acharya, B., & Hu, Y. C. "Deep learning-based parking occupancy detection framework using ResNet and VGG-16." Multimedia Tools and Applications 83.1 (2024): 1941-1964.

[110] Wan J, Yuan Y, Wang Q. "Traffic congestion analysis: A new perspective". In: 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), IEEE, pp 1398-1402. (2017).

[111] Shi X, Shan Z, Zhao N. "Learning for an aesthetic model for estimating the traffic state in the traffic video". Neurocomputing 181:29-37. (2016).

[112] Dubey, A., & Rane, S. "Implementation of an intelligent traffic control system and real time traffic statistics broadcasting". In: 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA), IEEE, pp 33-37. (2017).

[113] Rao, A., Phadnis, A., Patil, A., Rajput, T., & Futane, P. "Dynamic traffic system based on real time detection of traffic congestion". In: 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), IEEE, pp 1-5. (2018).

[114] Kulkarni, S., Ugale, S., Jawale, H., & Shinde, D. A. "Review on traffic congestion detection using image processing". Journal of Science & Technology (JST) 6(2):109-113. (2021).

[115] Eamthanakul B, Ketcham M, Chumuang N. "The traffic congestion investigating system by image processing from cctv camera". In: 2017 international conference on digital arts, media and technology (ICDAMT), IEEE, pp 240-245. (2017).

[116] Zhu F. "A video-based traffic congestion monitoring system using adaptive background subtraction". In: 2009 Second International Symposium on Electronic Commerce and Security, IEEE, pp 73-77. (2009).

[117] Xun, F. F., Yang, X. H., Xie, Y., & Wang, L. Y. "Congestion detection of urban intersections based on surveillance video". In: 2018 18th International Symposium on Communications and Information Technologies (ISCIT), IEEE, pp 495-498. (2018).

[118] Harris, C., & Stephens, M. "combined corner and edge detector". In: Alvey vision conference, Citeseer, pp 10-5244. (1988).

[119] Lucas BD, Kanade T. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: IJCAI'81: 7th international joint conference on Artificial intelligence, Vancouver, Canada, pp 674- 679. (1981).

[120] Perkasa O, Widyantoro DH. "Video-based system development for automatic traffic monitoring". In:2014 International Conference on Electrical Engineering and Computer Science (ICEECS), IEEE, pp 240-244. (2014).

[121] Lam CT, Gao H, Ng B. "A real-time traffic congestion detection system using on-line images". In: 2017 IEEE 17th International Conference on Communication Technology (ICCT), IEEE, pp 1548-1552. (2017).

[122] Amruta K Dhayafule and SR Gengaje. "Road congestion detection using image texture analysis". 2021.

[123] Haralick, R. M., Shanmugam, K., & Dinstein, I. (1973). "Textural Features for Image Classification." IEEE Transactions on Systems, Man, and Cybernetics, SMC-3(6), 610–621.

[124] Mu K, Hui F, Zhao X. "Multiple vehicle detection and tracking in highway traffic surveillance video based on sift feature matching". Journal of Information Processing Systems 12(2). (2016).

[125] Lowe DG. "Distinctive image features from scale-invariant keypoints". International journal of computer vision 60:91-110. (2004).

[126] Nguyen, H. N., Krishnakumari, P., Vu, H. L., & Van Lint, H. "Traffic congestion pattern classification using multi-class svm". In: 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), IEEE, pp 1059-1064. (2016).

[127] Bay, Herbert, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features." European conference on computer vision. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.

[128] Viola P, Jones M. "Rapid object detection using a boosted cascade of simple features". In: Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition. CVPR 2001, Ieee, pp I-I. (2001).

[129] Choudhury S, Chattopadhyay SP, Hazra TK. "Vehicle detection and counting using haar feature-based classifier". In: 2017 8th annual industrial automation and electromechanical engineering conference (IEMECON), IEEE, pp 106-109. (2017).

[130] Asmaa, Ouessai, Keche Mokhtar, and Ouamri Abdelaziz. "Road traffic density estimation using microscopic and macroscopic parameters." Image and Vision Computing 31.11 (2013): 887-894.

[131] Chan, Antoni B., and Nuno Vasconcelos. "Classification and retrieval of traffic video using auto-regressive stochastic processes." IEEE Proceedings. Intelligent Vehicles Symposium, 2005.. IEEE, 2005.

[132] Derpanis, Konstantinos G., and Richard P. Wildes. "Classification of traffic video based on a spatiotemporal orientation analysis." 2011 IEEE workshop on applications of computer vision (WACV). IEEE, 2011.

[133] Gonçalves, Wesley Nunes, Bruno Brandoli Machado, and Odemir Martinez Bruno. "A complex network approach for dynamic texture recognition." Neurocomputing 153 (2015): 211-220.

[134] Riaz, Amina, and Shoab A. Khan. "Traffic congestion classification using motion vector statistical features." Sixth International Conference on Machine Vision (ICMV 2013). Vol. 9067. SPIE, 2013.

[135] Andrews Sobral, Luciano Oliveira, Leizer Schnitman, and Felippe De Souza. "Highway traffic congestion classification using holistic properties." 10th IASTED international conference on signal processing, pattern recognition and applications. 2013.

[136] Luo, Z., Jodoin, P. M., Li, S. Z., & Su, S. Z. "Traffic analysis without motion features." 2015 IEEE international conference on image processing (ICIP). IEEE, 2015.

[137] Chaudhary S, Indu S, Chaudhury S. "Video-based road traffic monitoring and prediction using dynamic bayesian networks". IET Intelligent Transport Systems 12(3):169-176. (2018).

[138] Impedovo, D., Balducci, F., Dentamaro, V., & Pirlo, G. "Vehicular traffic congestion classification by visual features and deep learning approaches: a comparison." Sensors 19.23 (2019): 5213.

[139] Wang, P., Hao, W., Sun, Z., Wang, S., Tan, E., Li, L., & Jin, Y. "Regional detection of traffic congestion using in a large-scale surveillance system via deep residual TrafficNet." IEEE Access 6 (2018): 68910-68919.

[140] Kurniawan, J.; Syahra, S.G.; Dewa, C.K. "Traffic Congestion Detection: Learning from CCTV Monitoring Images using Convolutional Neural Network". Procedia Comput. Sci. 2018, 144, 291–297.

[141] Luo, Z., Jodoin, P. M., Su, S. Z., Li, S. Z., & Larochelle, H. "Traffic analytics with low-frame-rate videos." IEEE Transactions on Circuits and Systems for Video Technology 28.4 (2016): 878-891.

[142] Ribeiro, Matheus Vieira Lessa, Jorge Leonid Aching Samatelo, and Ana Lucia Cetertich Bazzan. "A new microscopic approach to traffic flow classification using a convolutional neural network object detector and a multi-tracker algorithm." IEEE Transactions on Intelligent Transportation Systems 23.4 (2020): 3797-3801.

[143] Wang, P., Li, L., Jin, Y., & Wang, G. "Detection of unwanted traffic congestion based on existing surveillance system using in freeway via a CNN-architecture trafficnet." 2018 13th IEEE conference on industrial electronics and applications (ICIEA). IEEE, 2018.

[144] Zhuo, Y., Wang, P., Sun, J., & Jin, Y. "Traffic congestion detection based on the image classification with cnn." Advances in Guidance, Navigation and Control: Proceedings of 2020 International Conference on Guidance, Navigation and Control, ICGNC 2020, Tianjin, China, October 23–25, 2020. Singapore: Springer Singapore, 2021.

[145] Adnan, M. F., Ahmed, N., Ishraque, I., Al Amin, M. S., & Hasan, M. S. "Traffic congestion prediction using deep convolutional neural networks: A color-coding approach." 2022 International Conference on Engineering and Emerging Technologies (ICEET). IEEE, 2022.

[146] Jiang, P., Ergu, D., Liu, F., Cai, Y., & Ma, B. "A Review of Yolo algorithm developments." Procedia computer science 199 (2022): 1066-1073.

[147] Nguyen, T. T., Calvert, S. C., Li, G., & van Lint, H. "Interpretable representation and customizable retrieval of traffic congestion patterns using causal graph-based feature associations". Data Science for Transportation 6(3):18. (2024).

[148] Jiang, S., Feng, Y., Zhang, W., Liao, X., Dai, X., & Onasanya, B. O. "A new multi-branch convolutional neural network and feature map extraction method for traffic congestion detection". Sensors 24(13):4272. (2024).

[149] Azfar, T., Li, J., Yu, H., Cheu, R. L., Lv, Y., & Ke, R. "Deep learning-based computer vision methods for complex traffic environments perception: A review". Data Science for Transportation 6(1):1. (2024).

[150] Khalladi, S. A., Ouessai, A., Benamara, N. K., & Keche, M. "Efficient road traffic video congestion classification based on the multi-head self-attention vision transformer model." Transport and Telecommunication 25.1 (2024): 20-30.

[151] Chan, Antoni B., and Nuno Vasconcelos. "Efficient computation of the kl divergence between dynamic textures." Technical Report (2004).

[152] Lea, C., Flynn, M. D., Vidal, R., Reiter, A., & Hager, G. D. "Temporal convolutional networks for action segmentation and detection." proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. 2017.

[153] Oord, A. V. D., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... & Kavukcuoglu, K. "Wavenet: A generative model for raw audio." arXiv preprint arXiv:1609.03499 (2016).

[154] Kalchbrenner, N., Espeholt, L., Simonyan, K., Oord, A. V. D., Graves, A., & Kavukcuoglu, K."Neural machine translation in linear time." arXiv preprint arXiv:1610.10099 (2016).

[155] https://keras.io/search.html?q=data+augmentation+

[156] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z."Rethinking the inception architecture for computer vision." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[157] Qiang, Zhenping, Libo He, and Fei Dai. "Identification of plant leaf diseases based on inception V3 transfer learning and fine-tuning." International Conference on Smart City and Informatization. Singapore: Springer Singapore, 2019.

[158] Shorten, Connor, and Taghi M. Khoshgoftaar. "A survey on image data augmentation for deep learning." Journal of big data 6.1 (2019): 1-48.

[159] HILMAN, Muhammad. "Multi object detection and tracking using optical flow density–Hungarian Kalman filter (Ofd-Hkf) algorithm for vehicle counting." Jurnal Ilmu Komputer dan Informasi (2018).

[160] Murugan, V., V. R. Vijaykumar, and A. Nidhila. "A deep learning RCNN approach for vehicle recognition in traffic surveillance system." 2019 international conference on communication and signal processing (ICCSP). IEEE, 2019.

[161] Xu, Bin, Bin Wang, and Yinjuan Gu. "Vehicle detection in aerial images using modified YOLO." 2019 IEEE 19th International Conference on Communication Technology (ICCT). IEEE, 2019.

[162] Chetouane, A., Mabrouk, S., Jemili, I., & Mosbah, M. "Vision-based vehicle detection for road traffic congestion classification." Concurrency and Computation: Practice and Experience 34.7 (2022): e5983.

[163] Jocher, G., Chaurasia, A. & Qiu, J. (2023). Ultralytics yolov8.URL: https://github.com/ultralytics/ultralytics

[164] Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. "Microsoft coco: Common objects in context." European conference on computer vision. Cham: Springer International Publishing, 2014.

[165] Ultralytics (2024) Yolo models. URL https://docs.ultralytics.com/models/

[166] Hendrycks, Dan, and Kevin Gimpel. "Gaussian error linear units (gelus)." arXiv preprint arXiv:1606.08415 (2016).

[167] Lu, Jia, and Li Cao. "Congestion evaluation from traffic flow information based on fuzzy logic." Proceedings of the 2003 IEEE international conference on intelligent transportation systems. Vol. 1. IEEE, 2003.

[168] Ali, Khalfi, and Guerroumi Mohamed. "Roadside parking spaces image classification using deep learning." International Conference on Computing Systems and Applications. Cham: Springer International Publishing, 2020.

[169] Khalfi, Ali, and Mohamed Guerroumi. "Transfer learning with image data augmentation for parking occupancy detection." 2023 International Conference on Advances in Electronics, Control and Communication Systems (ICAECCS). IEEE, 2023.

[170] Khalfi, A., Guerroumi, M., Haid, T., & Laradji, N. "Vision transformer for detecting traffic congestion using image mapping". Advances in Transportation Studies, 66. 2025.

[171] Khalfi ali , Guerroumi Mohamed et Zekai Şen, “Fuzzy logic multi-module for real time traffic congestion”. Journal of Network and Systems Management (under review)