Machine learning for road active safety in vehicular networks

Jialin Hao

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2024IPPAS003

Thèse de doctorat

MACHINE LEARNING FOR ROAD ACTIVE SAFETY IN VEHICULAR NETWORKS

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom SudParis

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (EDIPP)
Spécialité de doctorat : Signal, Images, Automatique et robotique

Thèse présentée et soutenue à Palaiseau, le 26/02/2024, par

JIALIN HAO

Composition du Jury :

Pascal Lorenz Professeur, Université de Haute Alsace, Colmar	Président
Khaled Boussetta Professeur, Paris 13	Rapporteur
Salah-Eddine El Ayoubi Professeur, CentraleSupelec	Rapporteur
Hadji Makhlouf Chercheur scientifique, Institut de Recherche Technologique SystemX	Examinateur
Thi-Mai-Trang Nguyen Professeur, Université Sorbonne Paris Nord	Examinatrice
Djamal Zeghlache Professeur, Télécom SudParis	Directeur de thèse
Rola Naja HDR et Associate Professor, ECE Paris Lyon - Ecole d'ingénieurs	Co-encadrant
Samir Tohmé Professeur, Université de Versailles Saint-Quentin-en-Yvelines	Invité

Acknowledgements

Three and a half years of doctoral study is gradually coming to an end. Looking back, everything I have experienced is vivid in my mind, and it will definitely leave a mark in the book of my life. Throughout the writing of this dissertation I have received a great deal of support and assistance.

Thanks to my supervisor Djamal, who not only protected me on the road of scientific research, but also taught me the principles of dealing with others. You made me realize that I should not lower my requirements for myself even when I was at a low point; you gave me guidance when I was anxious and confused; and you never hesitated to share your opinions with me. The words of you, for me who had just entered the gate of scientific research world, every sentence was valuable to me and brought me new understanding.

Thank you, my supervisor Rola, for guiding me step by step onto the path of doctoral research. Despite the pandemic preventing us from meeting face-to-face during my first year, you deeply understood my anxieties and consistently encouraged me to maintain confidence. In our subsequent collaborations, you never hesitated to praise and comfort me, nor did you shy away from addressing my shortcomings. Working with you has provided me with a clearer understanding of the field of research. Through several meaningful conversations, you generously offered me reassurance and guidance, sharing your own life experiences, which filled me with warmth.

Thanks to the reviewers of my thesis. Thanks for your precious time spent for reading my work and your suggestions to improve the presentation, as well as your recognition of my work.

I would particularly like to acknowledge Mr. Samir Tohmé, the DIGICOSME project, Laboratory SAMOVAR and Télécom SudParis. Who provide the chance and availability for me to pursue this thesis.

Thanks to all my anonymous papers reviewers, including but not limited to ICC, WiMob, PIMRC, WCNC, CICOM and Computer Networks. You have made improvements to my papers, which has broadened my horizons and given me the opportunity to improve my editing and their quality. In any case, every review brings me new progress.

Thanks to my dear parents, who have given me ample care and understanding since I was a child, and always give me the confidence to move forward. Thanks to my twin sister, like whom having one person in life is enough. Thank you to all my family members for making me feel loved and worthy of love, and for making me realize the infinite possibilities of life's journey.

Thank you to my boyfriend, Wei Huang. You have taught me how to love myself and understand how to love others. Love is selfless giving and acceptance, it is understanding and supporting

each other. It is enduring change and growth, intimacy and relaxation, and above all, it is the source of happiness. It is mutual respect, complementing each other's strengths, it is walking side by side.

Thanks to my two lovely cats, Cola and Mi, who always bring surprises and warmth to my life. The three-year bond has made us a family, and we will never leave each other.

I would like to thank my friends who have returned to China to work, those who are studying for Ph.D.s in foreign countries, those who are working in France, and those who have stayed in China. Despite our infrequent gatherings in recent years and the vast distances between us, our hearts remain connected, and our friendship endures.

Thank you to the companions who once shared time with me in France. With your companionship, I never felt lonely.

Thank you to those who have hurt me in the past, for you have not killed me but made me stronger. In my twenties, encountering you has taught me not to be reckless anymore.

Graduating a PhD is not the end, but a new beginning. I will take the aura and darkness of the past with me to the next journey full of hope.

With this thesis, I express my gratitude to all the friends, teachers and relatives who have appeared in my life!

致谢

2020 年 10 月，我刚满二十四岁，正处于心智幼稚与成熟的分水岭。在这个转折点上，我踏上了巴黎理工学院的征程，开启了一段全新的人生旅程。随着三年半的博士求学生涯逐渐接近尾声，回首往事，每一个经历都历历在目，定将在我的人生篇章中留下浓墨重彩的一笔。

首先，我要感谢我的导师 Djamal。你不仅在科研道路上为我提供了指导和支持，还教会了我待人处事的道理。你让我意识到，即使在低谷时期也不能放弃对自己的要求；在我迷失焦虑时给予我指引；并且毫不吝啬地与我分享你的见解。对于一个初入科研圈的我来说，导师的每一句话都是新的启示。

感谢我的导师 Rola，是你一步步引领我走上博士研究之路。尽管由于疫情，我们在博士第一年从未见过面，但你深深理解我的焦虑，始终鼓励我保持信心。在后续的合作中，你从不吝赞美与安慰，对我的不足也决不含糊。与你的合作让我对科研领域有了更清晰的认识。在几次深入的交谈中，你热心地给予我安慰和指导，向我分享你的人生经验，使我倍感温暖。

特别感谢我的论文审稿人。感谢您花费宝贵的时间阅读我的文章，以及提出的改进建议。您对我的工作的认可，我深表感激。

我还要特别感谢 Samir Tohmé 先生、DIGICOSME 项目、SAMOVAR 实验室和 Télécom SudParis，南巴黎电信学校。是你们为我提供了这个博士课题和研究环境，让我有机会踏上读博之路。

同时，感谢所有曾审阅过我投稿论文的审稿人，包括但不限于 ICC、WiMob、PIMRC、WCNC、CICOM 和 Computer Networks。你们提出的改进建议，无论大小，都拓展了我的视野，让我对研究课题有了更全面的认识。每一次审稿意见都是我进步的动力。

感谢我最亲爱的父母，你们从小给予我充分的关爱和理解，永远是我向前走的动力。感谢我的双胞胎姐姐，人生有一人如此，夫复何求？此外，感谢我的所有家人，你们让我感受到自己是被爱的，也让我认识到人生旅途的无限可能。

感谢我的男友，黄炜，你教会了我如何爱自己和爱别人。爱是付出和包容，爱是理解，爱是改变。爱是亲密，是放松，是快乐，是相互尊重、各有所长、并肩同行。

还要感谢我的两只可爱的猫，可乐和小咪。你们给我的生活带来了无尽的欢乐和温暖。三年的相处已经让我们成为了一家人，从选择你们的那一刻开始，我们就注定永远相伴，不离不弃。

感谢我已回国工作的、在异国求学的、在法国工作的、以及一直留在国内的朋友们。即使我们近年少有相聚，即使相隔遥远的距离，但我们心系彼此，友情依旧。

感谢曾经与我在法国一同度过时光的伙伴们，有了你们的陪伴，我从未感到孤独。

最后，我要感谢曾经伤害过我的人，杀不死我的让我更强大。在二十多岁的年纪里，因为遇到了你们，我已不再轻率。

博士毕业并不是终点，而是新的起点。我将带着过去的光辉和遗憾，踏上下一段充满希望的旅程。

最后，我要在此论文中向所有在我生命中出现过的朋友、老师和亲人们表示最诚挚的谢意！

Abstract

Lane change maneuvers are a significant contributor to road accidents, demanding effective solutions within vehicular networks. Current Lane Change Assist (LCA) strategies relying solely on Deep Reinforcement Learning (DRL) are limited by local vehicle information, overlooking a global view of traffic conditions. To address this, Unmanned Aerial Vehicles (UAVs), or drones, present a promising extension to vehicular network services due to their mobility, computational capabilities, and Line-of-Sight (LoS) communication links with road vehicles. This thesis focuses on developing a safe and efficient LCA maneuver within the context of Drone-Assisted Vehicular Networks (DAVN).

In the first step, we conduct a literature review on LCA within DAVN, highlighting the potential of drones to enhance road safety. Existing LCA approaches predominantly rely on local vehicle information and fail to consider overall traffic states. To address this limitation, we propose the GL-DEAR: joint global and local drone-assisted lane change platform based on Deep-Q Network (DQN) with a dynamic reward function, for LCA with drones' assistance.

The proposed platform consists of three modules: road with random risks and emergency vehicles; data file acquisition and processing; and real-time lane change decision-making. The lane change maneuver is based on a Deep Q-Network with dynamic reward functions. Specifically, we adopt the authentic NGSIM dataset-based lane change models for ordinary road vehicles to recreate real world lane change behaviors in the simulations. Numerical results demonstrate the platform's ability to achieve collision-free trips on risky highways with emergency vehicles.

In the second step, we identify a lack of calibration for the global update frequency in Federated Learning (FL) algorithms and the absence of thorough drone-level processing delay assessment. To this end, we propose the drone assisted Federated Reinforcement Learning (FRL)-based LCA framework, DAFL. This framework enables cooperative learning between ego vehicles [2] by applying FL. It includes a client reputation-based global model aggregation algorithm and a comprehensive analysis of End-to-End (E2E) delay at the drone. Specifically, the global update frequency is dynamically adjusted according to road safety measurements and drone energy consumption, yielding efficient results in simulations.

In the third step, we devise the DOT-P algorithm for optimizing drone trajectories in dynamic vehicular networks. This algorithm aims to balance drone energy consumption and road safety.

[2] An ego vehicle is a vehicle implementing the proposed algorithm.

We provide a comprehensive state-of-the-art review of the existing drone trajectory planning techniques. Then, based on the vehicle E2E delay modeling and the drone energy consumption modeling in the second step, we train a Offline Reinforcement Learning (ORL) model to avoid power-consuming online training. Simulation results demonstrate a significant reduction in drone energy consumption and vehicle E2E delay using the trained model.

In summary, this thesis offers a comprehensive framework for drone-assisted lane change maneuvers, federated learning, and drone trajectory optimization in vehicular networks, demonstrating the potential to improve road safety and traffic efficiency.

Résumé

Cette thèse porte sur le développement d'une manœuvre d'aide au changement de voie (lane Change Assistance, LCA) sûre et efficace dans le contexte des réseaux de véhicules assistés par drones (Drone Assisted Vehicular Network, DAVN). En effet, les changements de voie contribuent de manière significative aux accidents de la route, nécessitant des solutions efficaces au sein des réseaux routiers. Les LCA stratégies actuelles établies sur l'apprentissage par renforcement profond (Deep Reinforcement Learning, DRL) sont limitées par les informations locales sur les véhicules, négligeant une vue globale, comme des conditions de circulation. Pour résoudre ce problème, les véhicules aériens sans pilote (Unmanned Aerial Vehicles, UAVs), ou drones, présentent une extension prometteuse des services de réseau automobile grâce à leur mobilité, capacités informatiques et liaisons de communication en visibilité directe (Line-if-Sight, LoS) avec les véhicules routiers.

Dans un premier temps, nous faisons une étude bibliographique sur LCA au sein du DAVN, mettant en évidence le potentiel des drones pour améliorer la sécurité routière. Les approches LCA existantes s'appuient principalement sur des informations locales sur les véhicules et ne prennent pas en compte l'état global du trafic. Afin de réduire cette limitation, nous proposons le GL-DEAR : joint global and local drone-assisted lane change platform based on Deep-Q Network (DQN) with a dynamic reward function, for LCA with drones' assistance.

La plateforme proposée se compose de trois modules : route à risques aléatoires et véhicules d'urgence ; acquisition et traitement des données ; prise de décision de changement de voie en temps réel. La manœuvre de changement de voie est basée sur un Deep Q-Network avec des fonctions de récompense dynamiques. Plus précisément, nous adoptons les modèles de changement de voie authentiques basés sur l'ensemble de données NGSIM pour les véhicules routiers ordinaires afin de recréer les comportements de changement de voie du monde réel dans les simulations. Les résultats numériques démontrent la capacité de la plateforme à réaliser des trajets sans collision sur des autoroutes à risque avec des véhicules d'urgence.

Dans un deuxième temps, nous identifions un manque de calibrage de la fréquence de mise à jour globale des algorithmes d'apprentissage fédéré (Federated Learning, FL) et l' absence d' évaluation approfondie du délai de traitement au niveau du drone. Nous proposons donc un cadre d'apprentissage par renforcement fédéré (FRL) assisté par drone, DAFL. Ce cadre permet un apprentissage coopératif entre les véhicules de l'ego en appliquant FL. Il comprend un algorithme d'agrégation de modèles global basé sur la réputation du client et une analyse complète du délai de bout en bout (End-to-End, E2E) au niveau du drone. Plus précisément, la fréquence globale de mise à jour est ajustée dynamiquement en fonction des mesures de sécurité

routière et de la consommation énergétique des drones, ce qui donne des résultats efficaces dans les simulations.

Dans la troisième étape, nous concevons l'algorithme DOP-T pour optimiser les trajectoires des drones dans les réseaux de véhicules dynamiques. Cet algorithme vise à équilibrer la consommation énergétique des drones et la sécurité routière. Nous fournissons un état de l' art complet des techniques existantes de planification de trajectoire de drones. Ensuite, sur la base de la modélisation du délai E2E du véhicule et de la modélisation de la consommation d'énergie du drone. Dans la seconde étape, nous formons un modèle d'apprentissage par renforcement hors ligne (Offline-Reinforcement Learning, ORL) pour éviter une formation en ligne consommatrice d'énergie. Les résultats de la simulation démontrent une réduction significative de la consommation d'énergie des drones et du délai E2E du véhicule à l'aide du modèle entraîné.

En résumé, cette thèse propose un cadre global pour les manœuvres de changement de voie assistées par des drones, l'apprentissage fédéré et l'optimisation de la trajectoire des drones dans les réseaux véhiculaires, démontrant ainsi le potentiel d'amélioration de la sécurité routière et de l'efficacité du trafic.

Acronyms

AI Artificial Intelligence. 7

ANN Artificial Neural Network. 8

BSs Base Stations. 18

CAVs Connected Autonomous Vehicles. 22

CMFL Communication-Mitigated Federated Learning. 24

CNN Convolutional Neural Network. 21, 81

D2D Drone-to-Drone. 90

D2E Drone-to-Ego. 38

D2I Drone-to-Infrastructure. 18

D2V Drone-to-Vehicle. 25, 55–57, 72, 90, 114

D3QN Double Dueling DQN. 22

DAFL Drone Assisted Federated Reinforcement Learning. xiii, xiv, 4, 6, 27, 28, 46–48, 58, 70, 72, 113, 114

DAVN Drone Assisted Vehicular Network. xiii, 2–4, 6, 15, 17–20, 28, 113, 115

DDPG Deep Deterministic Policy Gradient. 21, 26

DEAR Deep Q-Network with Dynamic Reward Function. 29, 39, 41, 43, 44

DNN Deep Neural Network. 39–41, 43, 44

DOT-P Drone Optimal Trajectory Prediction. v, 3, 5, 6, 112, 114, 115

DQN Deep Q-Network. 6, 9, 15, 21, 25, 36, 48, 49, 81, 114

DRL Deep Reinforcement Learning. xiii, 4, 6, 7, 15, 17, 20, 21, 25, 28, 29, 32, 45, 47, 81, 113

E2E End-to-End. 3–6, 47, 69, 72, 73, 103, 105, 109, 114, 115

FL Federated Learning. xvi, 2–4, 6, 7, 12–15, 17, 20, 24, 25, 27, 28, 45–47, 72, 113, 114, 116

FLGU Federated Learning Global Update. 48, 49, 51

FRL Federated Reinforcement Learning. 3, 4, 6, 7, 15, 17, 25, 26, 28, 113, 116

GCN Graph Convolutional Neural Network. 116

GCQ Graphic Convolution Q-network. 22

GL-DEAR Joint Global and Local Controlled Deep Q-Network with Dynamic Reward Function. xiii, 2, 4, 6, 24, 28–30, 32, 34, 39–45, 47, 113, 114

GPS Global Positioning System. 2, 73

HFL Horizontal Federated Learning. 13, 14

HFRL Horizontal Federated Reinforcement Learning. 15

IID Independent and Identically Distributed. 13

IoT Internet-of-Things. 26

IRL Inverse Reinforcement Learning. 81

ITS Intelligent Transportation Systems. 1

KNN K-Nearest Neighbors. 39–41, 43, 44

LCA Lane Change Assistance. xiii, 1–4, 6, 15, 17, 20, 23, 28–30, 38, 45, 46, 73, 113

LoS Line-of-Sight. 2, 18, 19, 56, 90, 113

LQR Linear Quadratic Regulator. 75

LR Logistic Regression. 8

MDP Markov Decision Process. 10, 21–23, 27

ML Machine Learning. xvi, 1, 2, 4, 7, 10, 12, 15, 17, 31, 113

MPC Model Predictive Control. 21, 22, 75

NGSIM Next Generation Simulation Vehicle Trajectories and Supporting Data. xiii, 2, 4, 6, 29–32, 34, 35, 38, 45, 114

NLoS No Line-of-Sight. 56

NN Neural Network. 9, 11, 12, 76, 77, 81, 84

OARL Observation Adversarial Reinforcement Learning. 23

OBUs On-Board-Units. 2, 17

OP Optimization Problem. 20, 22

ORL Offline Reinforcement Learning. xviii, 3–6, 73, 82–84, 86, 101, 112, 115

PCA Principal Component Analysis. xiii, 30–32, 39, 43–45, 114

PG Policy Gradient. 39–41, 43–45

PLAS Policy in the Latent Action Space. 103, 105, 109, 112, 115

RL Reinforcement Learning. xviii, 2, 6, 7, 9, 10, 15, 29, 38, 81–84, 101, 103, 113

RN Relying Node. 18, 19

RNN Recurrent Neural Network. 81

RRAN Remote Radio Access Node. 18, 19

RSUs Road-Side-Units. 18

SGD Stochastic Gradient Descent. 11, 12

SNR Signal-to-Noise Ratio. 57

SUMO Simulation of Urban MObility. 20, 38, 39, 58, 59, 91

SVM Support Vector Machine. 8

TraCI Traffic Control Interface. 39

UAV Unmanned Aerial Vehicle. xv, xvii, xviii, 1–4, 17, 20, 25–27, 46, 52, 55–57, 70, 73–78, 81–86, 89–92, 94–96, 98, 113

ULCR Urgent Lane Change Request. 4, 24, 29, 31, 34, 47, 49, 51

V2D Vehicle-to-Drone. 2–6, 18, 38, 72, 73, 89, 105, 109, 115

V2I Vehicle-to-Infrastructure. 17, 18, 26

V2V Vehicle-to-Vehicle. 2, 17–19, 26

VANETs Vehicular Ad-Hoc Networks. 1, 20

VFL Vertical Federated Learning. 13, 14

VFRL Vertical Federated Reinforcement Learning. 15

XGBoost Extreme Gradient Boosting. 35, 114

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivations

Autonomous vehicles, or self-driving vehicles are able to perform intelligent driving controls without a human driver. The intelligent decisions are made in real time owing to the high-performance sensors such as Radar (Radio Detection and Ranging), LiDAR (Light Detection and Ranging) and infrared camera on the vehicles. Examples of the existing autonomous driving techniques are Lane Change Assistance (LCA), adaptive cruise control, auto parking assist, autonomous emergency braking and traffic jam assistant. These techniques enable to improve the driving experience.

Among various techniques in autonomous vehicle networks, LCA is an important research direction to tackle. In fact, changing lanes in an unsafe manner is one of the main causes of accidents. According to the National Highway Traffic Safety Administration (NHTSA), at least 33% of all crashes happen when vehicles change lanes or veer off the road [1]. It should be pointed out that lane change is a time-stringent process where the lane change decision should be made in 3–5 seconds. This issue yields that the driver should perceive and assess the surrounding environment, analyze the information and make the appropriate lane change decision in a tight time window. Otherwise, the safety of the driver or the smooth completion of the trip will be at risk. Thus, the development of a LCA platform, based on real-time data processing, is of paramount importance.

This thesis tackles the aforementioned problem using Machine Learning (ML) techniques. In fact, vehicular networks enable information exchange among vehicles, other end devices and public networks. This data exchange plays a key role in road safety, infotainment and traffic management devised for Intelligent Transportation Systems (ITS). Due to the ever-increasing demand of mobile services and the fast development of self-driving technologies, the data volume required, generated, collected, and transmitted by vehicular networks has seen an exponential escalation, which is known as big data. As a matter of fact, the efficient analysis of the generated vehicular data may reveal insightful patterns and correlations that may be exploited to make anticipated decisions. Thus, big data can provide valuable insights into vehicular networks, which can be employed to characterize and evaluate the performance of Vehicular Ad-Hoc Networks (VANETs), and design new protocols with big data intelligence.

On the other hand, drones or Unmanned Aerial Vehicles (UAVs) are considered as an important

extension of the vehicular networks. With drone's Line-of-Sight (LoS) links and their dynamic deployment ability, various tasks can be realized, such as drone assisted safety message broadcast, drone assisted ubiquitous internet access, drone assisted transportation surveillance and drone assisted additional spectrum provision [2]. In this context, we propose to tackle the enhancement of road active safety in Drone Assisted Vehicular Network (DAVN)s.

Actually, with Vehicle-to-Vehicle (V2V) communications, On-Board-Units (OBUs) may have access to a limited number of data observations that reflect a limited number of local kinematic samples. Hence, local datasets are unable to estimate the lane change maneuver at highway level. However, the drones can assist in gathering samples over the network at the cost of additional data exchange overheads through Vehicle-to-Drone (V2D) communications. Thus, the centralization of ML processing at the drone will yield satisfying results owing to the drones.

Indeed, OBUs may be reluctant to share their individual kinematic parameters and Global Positioning System (GPS) positions with the drone and other OBUs for security purpose and resource optimization; however, excessive data exchange may drain resources available for vehicular communications. This limitation may be fixed with a collaborative learning model that does not rely on sharing individual data sets. Therefore, Federated Learning (FL) is proposed as a decentralized learning technique where training datasets are unevenly distributed over learners, instead of centralizing all the data [3]. FL is one instance of the more general approach of "bringing the code to the data, instead of the data to the code" and addresses the fundamental problems of privacy, latency and locality of data.

In the first step, we are motivated by developing a LCA platform for DAVN, called GL-DEAR: Joint Global and Local Controlled Deep Q-Network with Dynamic Reward Function. This platform includes a Reinforcement Learning (RL) agent with a dynamic reward function considering safety, comfort and efficiency perspectives. Based on large-scale traffic information and the assistance from drones, the platform is able to detect road anomalies and emergency vehicles, and reduce the vehicle's total travel time and road collision rate.

Alternatively, mobility modeling is an important pillar in vehicular networks that plays a vital role in the performance evaluation. In fact, authentic mobility models reflect the movement patterns of vehicles on the road. They should derive a movement pattern in such a way that the generated pattern reflects real-world behaviors of vehicles on the road. This leads us to orient our efforts towards extracting real mobility models from the authentic Next Generation Simulation Vehicle Trajectories and Supporting Data (NGSIM) Vehicle Trajectories and Supporting Data [4] to recreate the vehicle mobility on real highways. Through the analysis of the NGSIM datasets of vehicle mobility, an amount of valuable information can be obtained, such as the practical mobility model, network connectivity, spatial and temporal density distribution.

It should be pointed out that, the use of battery-powered UAVs greatly relies on the limited battery life. To this end, researches have been conducted on this topic. Among them, energy harvesting provides a promising way to realize self-powered UAVs. According to [5], solar energy harvesting for UAVs mainly relies on photovoltaic cells and can reach watt-scale output power. Meanwhile, mechanical energy harvesting for UAVs can be further refined to wind-induced vibration and flapping wing motion whose output power is in the milliwatt scale. Reference [6]

proves the possibility of harvesting vibration and solar energy in a mini UAV to increase its endurance without adding significant mass or the need to increase the size of the fuel system. Experiments using an remote controlled glider aircraft with a 1.8m wing span show that the solar panels could charge a 170 mAh battery to 14% capacity and the piezoelectric patches could charge the EH300 4.6 mJ internal capacitor to 70% capacity during a 13-minute flight.

Moreover, in a DAVN, a huge number of vehicle requests will be sent to drones, leading to non-negligible queuing delay at the drone side. In this context, reducing the queuing delay for end users, and energy consumption for drones is a challenging task. Consequently, as the second contribution, we propose a FL algorithm that dynamically adjusts the global update frequency of the drone. In fact, higher global update frequency leads to more updated lane change model at the expense of higher communication overhead; while lower global update frequency leads to lower communication cost but outdated lane change model. Using dynamic adjustments, we are able to achieve the best trade-off between communication cost and road safety.

In more detail, we perform an accurate mathematical modeling of the energy consumption for the drones that play the role of central server. Besides, we also conduct a detailed analysis for the End-to-End (E2E) V2D communication delay based on a M/G/1 priority queue with preemption at the drone side. Then, a dynamic frequency adaptation framework is proposed, where the global update frequency is dynamically adjusted according to drone energy consumption and vehicle communication delay.

In the previous work, we assumed that the drones are statically located. In an attempt to generalize the study, we propose a Drone Optimal Trajectory Prediction (DOT-P) algorithm using Offline Reinforcement Learning (ORL) that dynamically deploys drones over the highway. The goal is to predict the optimal trajectories for the drone in order to reduce drone's energy consumption and the E2E communication delay. Specifically, ORL models avoid online training, which requires long flight for drones and thus huge energy consumption. Simulation results show that the trained model achieves the two previous goals.

The objectives and contribution of this thesis are detailed in the following section.

1.2 Objectives and contributions

Our thesis workflow and time allocation are presented in figure 1.1: The objectives and tasks are detailed as follows:

- **Objective 1** Data acquisition using drones and data processing. This objective consists of 3 tasks.
  Task 1 Study of the state of the art of DAVN architecture.
  Task 2 Clean acquired datasets.
  Task 3 Process datasets using dimension reduction techniques.

- **Objective 2** Critical real-time LCA using Federated Reinforcement Learning (FRL)
  Task 1 Literature survey of lane change maneuver.

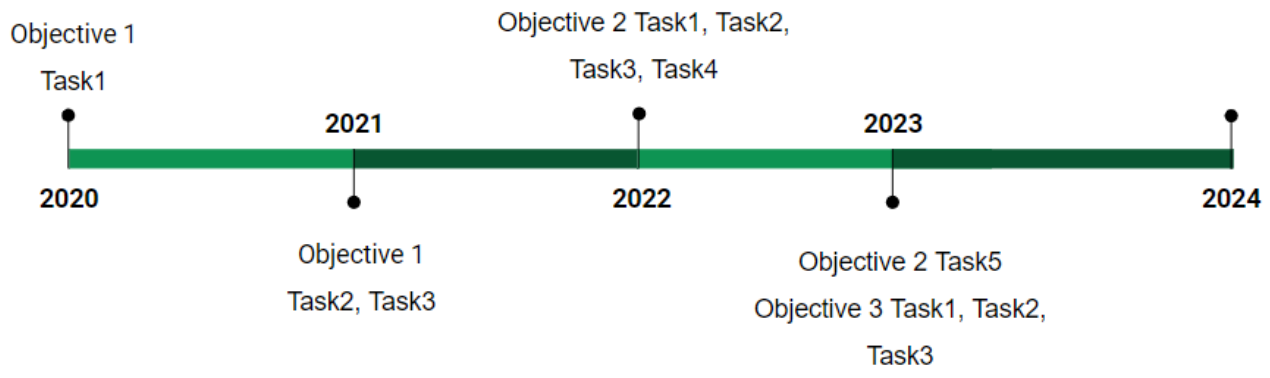

Figure 1.1: Thesis workflow chart and time allocation

Task 2 Proposition of a ML algorithm that efficiently assists the driver to make the lane change decision.
Task 3 Extraction of an authentic mobility model from NGSIM dataset.
Task 4 Simulation in order to evaluate the performance of the proposed algorithm.
Task 5 Enhancement of the LCA platform with FL.

- **Objective 3** Drone optimal trajectory prediction for energy and delay efficient LCA
  Task 1 Study of the state of UAV trajectory planning.
  Task 2 Design of an ORL algorithm for drone trajectory planning to improve drone energy consumption and V2D E2E delay.
  Task 3 Simulation in order to evaluate the performance of the proposed algorithm.

Our thesis addresses the improvement of road active safety in drone vehicular networks in the context of delay-stringent lane change services. More specifically, we achieved the following contributions:

- **Contribution 1** - We propose a LCA platform based on Deep Reinforcement Learning (DRL) for DAVN, GL-DEAR. The DRL agent possesses an comprehensive reward function considering road safety, travel efficiency and passenger's comfort perspectives. Specifically, a part of the reward is dynamically adjusted according to road safety performance.
- **Contribution 2** - We introduce drones to the proposed LCA platform where each drone plays three important roles in the lane change maneuver: 1) it provides global traffic information to the lane change agent; 2) it dynamically adjusts the collision reward function of the lane change agent; 3) it performs global control by sending an Urgent Lane Change Request (ULCR) to the corresponding vehicle.
- **Contribution 3** - We propose a FRL architecture in DAVN for LCA, called DAFL: Drone Assisted Federated Reinforcement Learning. In addition, we provide an accurate E2E communication delay modeling between the drone server and vehicle client based on M/G/1 priority queue with preemption, and the drone's energy consumption.
- **Contribution 4** - A dynamic frequency adaptation framework is proposed to achieve the

optimal trade-off between the road active safety performance and drone energy consumption, based on the derived models

- in the second step - We devise a drone optimal trajectory prediction algorithm based on ORL, called DOT-P. The algorithm is able to reduce V2D E2E communication delay and drone energy consumption by dynamically allocating drones over the highway.

1.3 Thesis context

This work is supported by Labex DigiCosme (project ANR11LABEX0045DIGICOSME) operated by ANR as part of the program *Investissement d'Avenir* Idex ParisSaclay (ANR11IDEX000302).

The doctoral project is being prepared, in accordance with the general conditions in force at Télécom SudParis, as part of enrollment in the doctoral program of the Institut Polytechnique de Paris for the academic years 2020 - 2024. The registered specialty is *Informatique, données, IA* in laboratory SAMOVAR.

Thesis director is Djamal ZEGHLACHE with 35% of the supervision proportion. Co-supervisors are Rola NAJA with 50% supervision proportion, and Samir TOHME with 15% supervision proportion.

1.4 List of publication

1. Jialin Hao, Rola Naja, Djamal Zeghlache, "Drone-Assisted Lane Change Maneuver using Reinforcement Learning with Dynamic Reward Function," IEEE International Conference on Wireless and Mobile Computing, Networking and Communications, October 2022, Greece
2. Jialin Hao, Rola Naja, Djamal Zeghlache, "Joint Local Reinforcement Learning Agent and Global Drone Cooperation for Collision-Free Lane Change," Springer EAI International Conference on Computational Intelligence and Communications, December 2022, Virtual
3. Scientific report, Jialin Hao. Dataset generation for drone optimal placement using machine learning. Telecom SudParis; Institut Polytechnique de Paris. 2022. hal-04192400
4. Jialin Hao, Rola Naja, Djamal Zeghlache, "GL-DEAR: Global Dynamic Drone Assisted Lane Change Maneuver for Risk Prevention and Collision Avoidance, "IEEE International Conference on Communications, May 2023, Italy
5. Poster, Jialin Hao, Rola Naja, Djamal Zeghlache. Machine learning for road active safety global dynamic drone assisted lane change maneuver for risk prevention and collision avoidance: Global dynamic drone assisted lane change maneuver for risk prevention and collision avoidance. SAMOVAR Ph.D Day 2023, May 2023, Evry, France. , 2023. hal-04221843
6. Jialin Hao, Rola Naja, Djamal Zeghlache, "Adaptive Federated Reinforcement Learn-

ing for Critical Realtime Communications in UAV Assisted Vehicular Networks," under revision, Computer Networks, September 2023

1.5 Thesis structure

This thesis is structured as follows. In chapter 2, we present the theoretical basis and state-of-the-art of machine learning techniques including supervised learning, unsupervised learning, reinforcement learning and FL. Particularly, we discuss FRL, which combines FL and RL.

Chapter 3 is devoted to LCA in DAVN. We first present the DAVN basis and shed the light on drone's potential of improving road active safety for vehicular networks. Then, we review the existing work of LCA maneuver using DRL and other techniques. Specifically, a detailed comparison table of different DRL-based LCA maneuvers is given. Afterwards, we study the related work about FL and FRL for DAVN. Based on the literature study, we are able to propose our LCA platform, GL-DEAR, in chapter 4, and the enhanced version of GL-DEAR using FL, DAFL, in chapter 5.

Chapter 4 introduces the three modules of the GL-DEAR platform. Then, the extraction of an authentic lane change model from the NGSIM dataset is explained. Next, the details of our GL-DEAR platform, which is a Deep Q-Network (DQN) with a dynamic reward function is highlighted.

The FRL based DAFL algorithm is detailed in chapter 5. Moreover, we provide an accurate mathematical modeling of drone energy consumption and V2D E2E communication delay based on M/G/1 queue. With the two models, we are able to define performance parameters to achieve the best trade-off between delay and energy consumption by dynamically adjusting the global update frequency of the DAFL.

Based on the mathematical modeling, in chapter 6, we study the drone optimal trajectory prediction problem. To this end, we first study the state-of-the-art of drone trajectory planning techniques. Then, we present our DOT-P model based on ORL in order to reduce drone energy consumption and the vehicle communication delay.

Finally, we conclude the thesis in chapter 7 with future perspectives.

Chapter 2

Theoretical Basis on Machine Learning

2.1 Introduction

In our thesis, we orient our efforts towards enhancing road active safety by proposing innovative platforms. These platforms rely on ML techniques. Thus, we found it necessary to dedicate this chapter to developing ML basic algorithms.

In fact, ML has gained a lot of attention in recent years. It is a branch of Artificial Intelligence (AI) and computer science which focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy. In this section, some basic ML algorithms are presented, including supervised learning, unsupervised learning, RL and FL.

Special attention is paid to FL, where an algorithm is trained across multiple decentralized edge devices without data exchanges. Moreover, we present FRL, where agents cooperate to learn a global model from the environment, each with their own RL model.

The organisation of this chapter is as follows: section 2.2 and 2.3 present supervised and unsupervised machine learning. Section 2.4 reviews RL, more specifically, DRL. Section 2.5 presents FL basis, including the original Federated Averaging algorithm, and the two categories of FL. In addition, we give a brief discussion about FRL in section 2.5.4. Finally, section 2.6 ends the chapter with a conclusion.

2.2 Supervised Learning

Supervised learning problems can be categorized into **regression** problems and **classification** problems. In a regression problem, the goal is to map continuous input to continuous output, while a classification problem aims to map discrete input into different categories, which means a discrete output [7].

A more formal description of supervised learning is: given a training set, learn a hypothesis function $h : X \to Y$ so that $h(x)$ is a "good" predictor for the corresponding value of y.

The accuracy of the hypothesis function is measured by a **cost function**, averaging the difference between each pair of prediction $h(x_i)$ and actual output y_i, as defined in equation 2.1.

The parameter vector θ of the function $h_\theta(x)$ is estimated by minimizing the cost so that the hypothesis function is accurate.

$$J(\theta_0, \theta_1) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{y}_i - y_i)^2 = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x_i) - y_i)^2 \tag{2.1}$$

The most often used method to estimate the parameters in the hypothesis function is **gradient descent**. The algorithm is described as follow:
Repeat until convergence:

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta) \tag{2.2}$$

where $j = 0, 1, ...n$ represents the feature index number; α is the learning rate.

A second way of minimizing J is called **normal equation**. It allows us to find the optimal θ without iteration. The normal equation formula is given in 2.3.

$$\theta = (X^T X)^{-1} X^T y \tag{2.3}$$

Some typical supervised learning models are:

- Linear regression: Linear regression is a method that adopts a linear approach to model the connection between a single numerical response variable and one or more predictor variables, which are also referred to as dependent and independent variables.
- Polynomial regression: Polynomial regression is a regression analysis technique that models the non-linear relation between the independent variable, x, and the dependent variable, y [8].
- Logistic Regression (LR): The logistic model, also known as the logit model, is a statistical framework used to represent the likelihood of an event occurring. It is achieved by expressing the natural logarithm of the odds of the event as a linear combination of one or more independent variables [9].
- Artificial Neural Network (ANN): ANNs are inspired by the organization of biological neural networks in animal brains. They consist of interconnected units called artificial neurons, organized into layers, each capable of specific operations on their inputs. Signals move from the input layer to the output layer, often passing through multiple intermediary layers. Typical uses of ANNs are function approximation, regression analysis, classification and data processing.
- Support Vector Machine (SVM): SVMs find extensive application in both classification and regression tasks. They demonstrate efficiency in achieving non-linear classification through a technique known as the kernel trick, which implicitly transforms their inputs into high-dimensional feature spaces [10].

2.3 Unsupervised Learning

Unsupervised learning or clustering allows the model to discover undetected patterns and information on its own. It mainly deals with unlabeled data.

Some typical unsupervised learning models are:

- K-means: K-means clustering is a method of vector quantization aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest cluster centroid, as illustrated in Fig. 2.1. Several use cases for k-means are like document classifications, delivery store optimization, identifying crime localities, customer segmentation and insurance fraud detection [11].

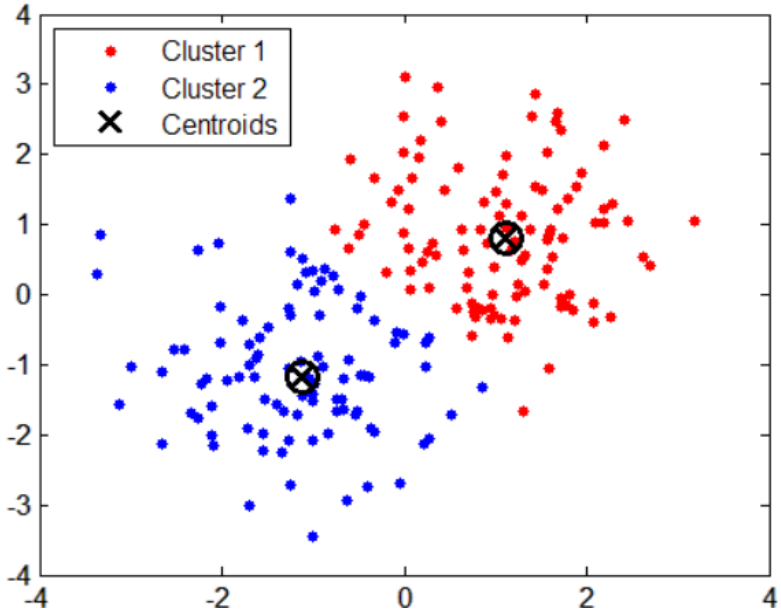

Figure 2.1: K-means: clusters and centroids

- Anomaly Detection: Anomaly detection is the process of identifying unexpected items or events in data sets, which differ from the norm. It is often applied on unlabeled data, which is known as unsupervised anomaly detection [12].
- Autoencoder: An autoencoder is a type of ANN used to learn efficient data codings in an unsupervised manner. The aim of an AE is to learn a representation (encoding) for a set of data, typically for dimension reduction [13]. More concretely, we'll design a Neural Network (NN) architecture such that we impose a bottleneck in the network which forces a compressed knowledge representation of the original input. A simple illustration is in Fig. 2.2.

2.4 Reinforcement Learning

In this section, we first give a brief introduction of RL. Then, a special attention will be paid to DQN, which will be used later in designing our proposed lane change maneuver.

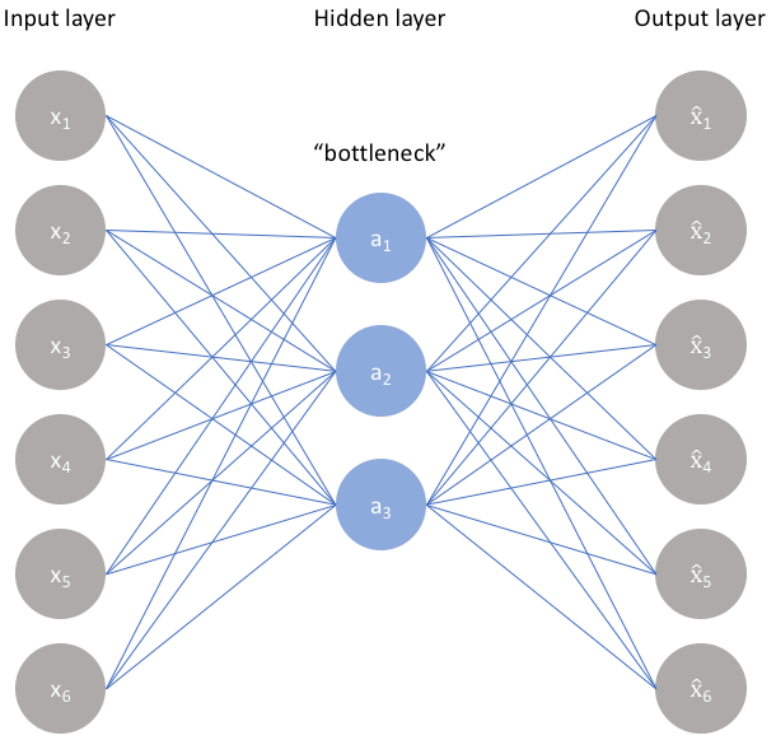

Figure 2.2: Autoencoder architecture. A bottleneck constrains the amount of information that can traverse the full network, forcing a learned compression of the input data.

RL is a branch of ML, where an agent acts in an environment and tries to learn a policy π to decide the action a to take, given a state s, while maximizing the cumulative reward function R.

The RL problem is often modeled as a Markov Decision Process (MDP), which is defined as the tuple $< S, A, T, R, \gamma >$, where S is the set of state, A is the set of actions, $T : S \times A \to S$ is the state transition probability function, $R : S \times A \times S \to \mathbb{R}$ is the reward function and $\gamma \in [0, 1]$ is a discount factor.

In RL, there are two categories of algorithms: value-based and the policy-based. In addition, there is also an actor-critic algorithm that can be obtained by combining the two.

Among the value-based algorithms, **Q-learning** is a typical widely-used one. It is also a model-free algorithm, where the optimal strategy is formulated by selecting the action with the highest Q value in each state. This strategy maximizes the expected return for all subsequent actions from the current state. The most important part of Q-learning is the update of Q value.

In more detail, in Q-learning, the agent tries to learn the optimal action value function, $Q^*(s, a)$. This function is defined as the maximum total reward, also called the Q-value, when being in a state, s, taking some action, a, and following the optimal policy, π^*.

$$Q^*(s, a) = R(s, a) + \gamma \max_a Q(s', a), \tag{2.4}$$

If the values of $Q(s', a)$ are known, the optimal policy is then to select the action a' that

maximizes the expected value of $Q(s', a)$. Further, $Q(s', a)$ depends on $Q(s'', a)$ which will then have a coefficient of γ^2. So, the Q-value depends on Q-values of future states as shown here:

$$Q(s, a) \rightarrow \gamma Q(s', a) + \gamma^2 Q(s'', a) + \cdots + \gamma^n Q(s^{'\cdots n}, a) \tag{2.5}$$

One can see that γ controls the contribution of rewards in the future.

In practical situations, this is implemented as an update:

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \underset{a}{max} Q(S_{t+1}, a) - Q(S_t, A_t) \right], \tag{2.6}$$

where α is the learning rate.

Mathematically, the state-action value function can be defined as

$$Q_\pi(s, a) \doteq \mathbb{E}_\pi \left[R_t | S_t = s, A_t = a \right] = \mathbb{E}_\pi \left[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s, A_t = a \right] \tag{2.7}$$

In **Deep Q-Learning**, a NN with weight θ is used as a function approximator of the optimal value function, i.e. $Q(s, a; \theta) \approx Q^*(s, a)$ is used to approximate the Q-value function. The state is given as the input and the Q-value of all possible actions is generated as the output [14].

The consists of following steps:

1. Store every past experience in memory
2. The next action is determined by the maximum output of the Q-network
3. The loss function is the mean-squared error of the predicted Q-value and the target Q-value, $Q^* = R_{t+1} + \gamma \underset{a}{max} Q(S_{t+1}, a)$ [15].

The network is then trained using Stochastic Gradient Descent (SGD) by minimizing the loss. The mini-batches with size M of experiences, described by the tuple $e_t = (s_t, a_t, r_t, s_{t+1})$, are drawn from the experience replay memory as indicated in step 1. This is called **experience reply**. The loss function at iteration i is defined as,

$$L_i(\theta_i) = \mathbb{E}_M \left[(r + \gamma \underset{a'}{max} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i))^2 \right] \tag{2.8}$$

Here, θ_i^- are the network parameters used to calculate the target at iteration i. In order to make the learning process more stable, we use two networks with the same architecture during the training process, one called the **prediction network**, the other called the **target network**. The parameters of the target network are held fixed for a number of iterations and then periodically updated with the latest version of the trained parameters of the prediction network. The trade-off between exploration and exploitation is handled by following an **ϵ-greedy policy**. This means that a random action is selected with probability ϵ, and otherwise the action with the highest value is chosen.

2.5 Federated Learning

The current trend in the usage of ML in vehicular networks focuses on centralized algorithms, where a powerful learning algorithm, often a NN, is trained on the massive dataset collected from the edge devices on the vehicles. Once the training is completed, the model parameters are sent back to the edge devices for prediction purposes, as shown in the left part of Fig. 2.3.

Nevertheless, the size of the generated data is huge when we need to build wider and deeper NN architectures for successful training. In this case, training a model with data transmission from the edge devices to the cloud center reliably may be too costly in terms of bandwidth, introduce unacceptable delays, and infringe user privacy [16]. FL, where each client trains a shared model with its own data and only communicate the update to a central server, is a promising method to the raised issues.

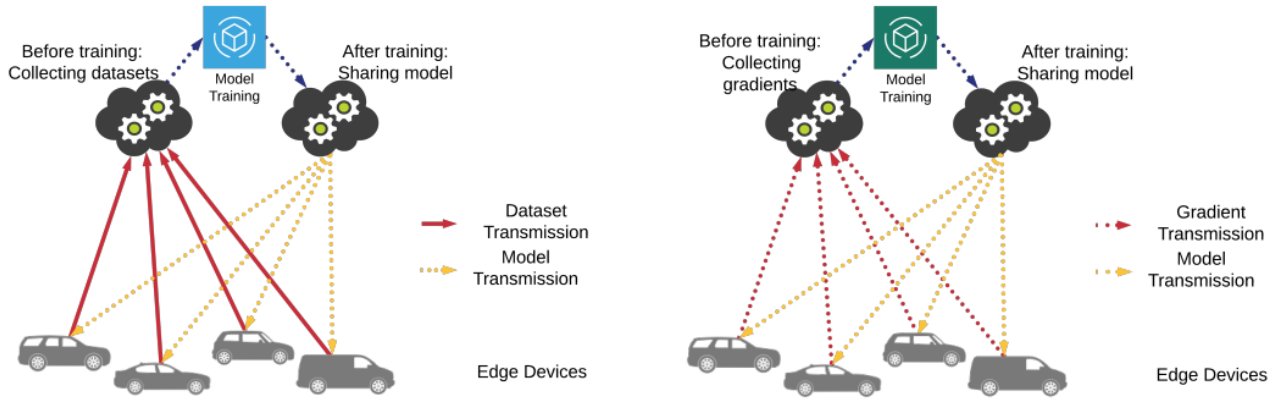

Figure 2.3: Left: Model training for ML in a vehicular network; Right: Model training for FL in a vehicular network

2.5.1 Federated averaging algorithm

The FL algorithm proposed in [17] is a decentralized approach, which consists of training an algorithm across multiple decentralized edge devices with their local data, without exchanging them. In other words, it leaves the training data distributed on the mobile devices and learns a shared model (e.g. a NN) by aggregating locally computed updates, as shown in the right part in Fig. 2.3. The pseudo code of the algorithm is shown in Algorithm. 1.

One can see that, the main algorithm consists of two parts: the server part and the client part, as described in Algorithm 1. At the server side, the server initializes the weight w_0 of the model (NN). Then for each communication round, the server chooses some of the clients (at least 1) with a fraction C and each client chosen will do local update with the ClientUpdate function simultaneously. Afterwards, clients will send their computation results to the server. Finally, the server will compute a global updated weight and send it back to clients; At the client side, for each client, the dataset is split into batches of size B. For each epoch and each batch, the clients update the local weight once using SGD with learning rate η.

The amount of computation is controlled by three key parameters: C, the fraction of clients that

Algorithm 1 `Federated averaging (FedAvg) algorithm` The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \cdots$ **do**
  $m \leftarrow max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  end for
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
end for

ClientUpdate(k, w)**:** ▷ Run on client k
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \bigtriangledown l(w; b)$
  end for
end for
return w to server

is chosen to perform computation each round; E, the number of training each client performs in each round; and B, the local mini-batch size used for the clients' updates. It can be noticed that FL is limited by the training time of the slowest participating devices, called stragglers [18].

The authors of [17] show, using experiments, that this approach is robust to the unbalanced and non-Independent and Identically Distributed (IID) data distribution, which is the common case in the real world.

2.5.2 Horizontal and vertical federated learning

According to different data features can sample identities, FL and further be divided into two categories: Horizontal Federated Learning (HFL) and Vertical Federated Learning (VFL).

HFL is suitable for situations where the data features of participants overlap more and the sample identities overlap less, for example, customer data of two banks in different regions. Since the data features of participants in HFL are aligned, HFL is also called feature-aligned FL. Figure 2.4 shows the horizontal partition of the data, where multiple rows of samples with the same characteristics from multiple participants are combined for FL. Consequently, HFL increases the total number of training samples [19].

VFL is suitable for situations where the participant training sample identities are more overlapped while the data features are less overlapped, for example, the common customer data of banks and e-commerce companies in the same region. As shown in Fig. 2.5 different data features of common samples from multiple participants are partitioned vertically and combined

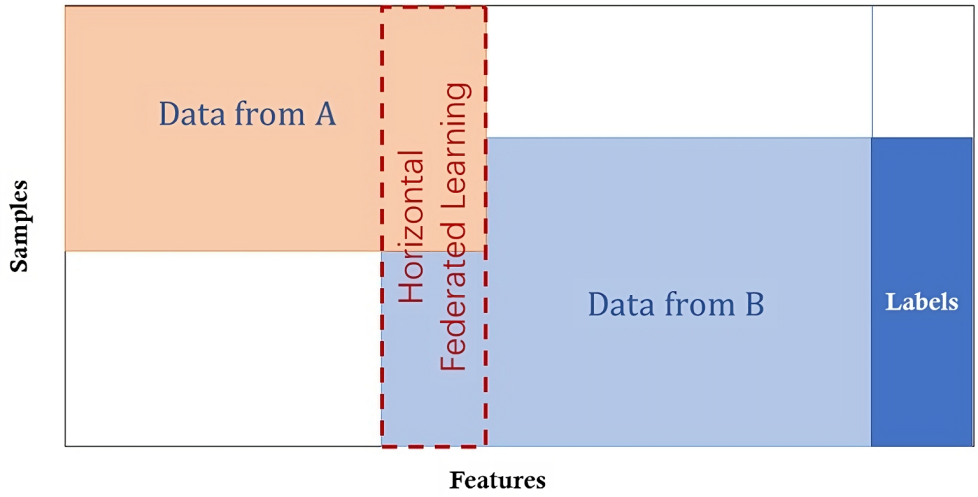

Figure 2.4: Horizontal federated learning [19]

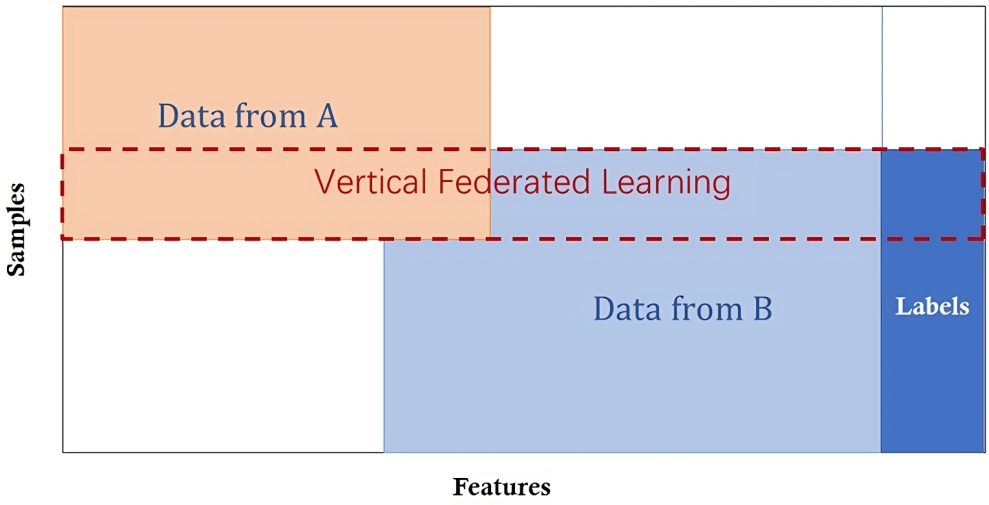

Figure 2.5: Vertical federated learning [19]

for FL. VFL is also called sample-aligned FL since the training samples of participants in VFL are aligned.

To summarize, the name of HFL comes from the "horizontal division" of training data, that is, the row (horizontal) division of the data matrix or table. Data in different rows have the same data characteristics, thus, the data characteristics are aligned; while the name of VFL comes from the "vertical division" of the training data, that is, the column (vertical) division of the data matrix or table. Data in different columns have the same sample ID, thus, the training samples are aligned.

2.5.3 Advantages and disadvantages of federated learning

FL offers several notable advantages and drawbacks in the realm of decentralized machine learning. On the positive side, it provides an effective data privacy solution by releasing the

need for client-side data to be transmitted to a central server; thereby addressing user concerns about data security. Moreover, this approach significantly reduces the computational burden on the server, as both gradient calculations during the training phase and model inference occur on the client-side. Additionally, FL results in lower online latency, as model inference happens locally, eliminating the need for constant server requests and averting delays arising from data network transfers.

However, there are associated challenges. Unstable client networks can pose difficulties in maintaining consistent model updates. Furthermore, client data may not be identically distributed, necessitating server-side techniques like Federated Stochastic Gradient Descent (FederatedSGD) to address data distribution disparities. Additionally, limited access to real data can make it challenging to perform accurate offline evaluations of model performance on the server. Lastly, tuning the model's architecture, hyperparameters, and data schema must be predefined, making post-deployment adjustments a complex endeavor.

2.5.4 Federated reinforcement learning

Local models in FRL are trained based on RL models. Consequently, FRL falls into two categories: Horizontal Federated Reinforcement Learning (HFRL) and Vertical Federated Reinforcement Learning (VFRL).

To provide further explanation, a comparison of HFRL and VFRL is shown in Fig. 2.6. In HFRL, the environment with which each agent interacts is distinct from the others, whereas the state space and action space of different agents are aligned to address similar issues. Multiple agents interacting with their own version of the environment can help accelerate training and improve model performance by sharing experience. Conversely with VFRL, multiple agents interact with a common global environment, yet each agent is limited in its ability to observe limited state information within its view. In this particular instance, the vertical arrangement of observations in VFRL presents a more intricate issue that has not been extensively examined in the existing literature [20].

2.6 Conclusion

In this chapter, we present a literature review about ML techniques. We start by introducing common supervised and unsupervised ML techniques. Then, we introduce RL and DRL, where DQN method is detailed, on which our proposed model is based. Furthermore, we discuss FL, including the original federated averaging algorithm, different FL categories, as well as the advantages and disadvantages of FL. Finally, we look into FRL, which is the combination of RL and FL.

Since our goal is to propose a real-time lane change maneuver, it is necessary to study the state-of-the-art of the lane change problem in vehicular networks. Thus, in the following chapter, we provide the bibliographic study on LCA in DAVN.

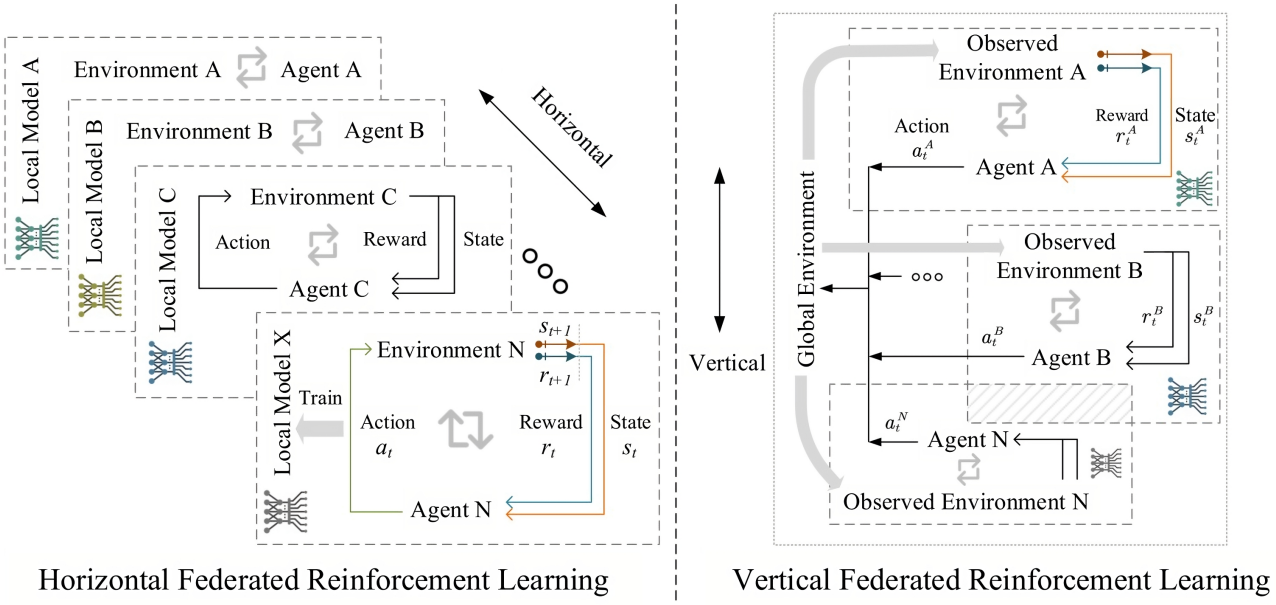

Figure 2.6: Comparison of horizontal federated reinforcement learning (HFRL) and vertical federated reinforcement learning (VFRL)

Chapter 3

Bibliographic Study for Lane Change Assistance in Drone Assisted Vehicular Networks

3.1 Introduction

Lane Change Assistance is crucial for road active safety. With drone's potential to assist autonomous driving, we believe that DAVN will play a key role in preventing collisions due to faulty lane change decisions. In this context, the development of a lane change algorithm based on ML techniques for DAVN is necessary in order to improve the road active safety. Thus, this chapter sheds the light on the state-of-the-art study performed in LCA related with DAVN.

The rest of this chapter is organised as follows: first, we introduce DAVN architecture, characteristics and services. Then, we review the existing LCA maneuvers based on DRL and other techniques. Especially, a detailed comparison table of different DRL-based LCA maneuvers are provided. Afterwards, we exhibit the research studies conducted for FL in DAVN, followed by a literature study of FRL in vehicular networks. Finally we end this chapter with a conclusion.

3.2 Drone-Assisted Vehicular Network

Drones, or UAVs, equipped with dedicated sensors or communication devices, have been considered to be an important extension of the vehicular network. This section presents the study of the DAVN including DAVN architecture, DAVN characteristics, DAVN advantages to the traditional vehicular networks, and related work.

3.2.1 DAVN architecture

A DAVN consists of vehicles, infrastructures and drones as explained here after [2] and as illustrated in Fig. 3.1:

- **Vehicle** Vehicles are embedded with OBUs to communicate with other network elements. It is to be noted that Vehicle-to-Vehicle (V2V) communication enables data transmission among vehicles; whereas, Vehicle-to-Infrastructure (V2I) communications handle data exchange between vehicle and infrastructure.

- **Infrastructure** Both RSUs and cellular Base Stations (BSs) are considered as infrastructure entities in DAVN.
- **Drone** Two kinds of drones are considered in DAVN: Relying Node (RN) drones and Remote Radio Access Node (RRAN) drones, as shown in Fig. 3.1. RN drones can be treated as flying vehicular nodes. They relay data for V2V communications and access infrastructures in the same way as vehicles. While RRAN drones perform as remote radio access points that can be dynamically allocated to required positions to assist V2I data exchanges: vehicles exchange data with drones through the V2D communications and drones communicate the data to infrastructure through the Drone-to-Infrastructure (D2I) communications.

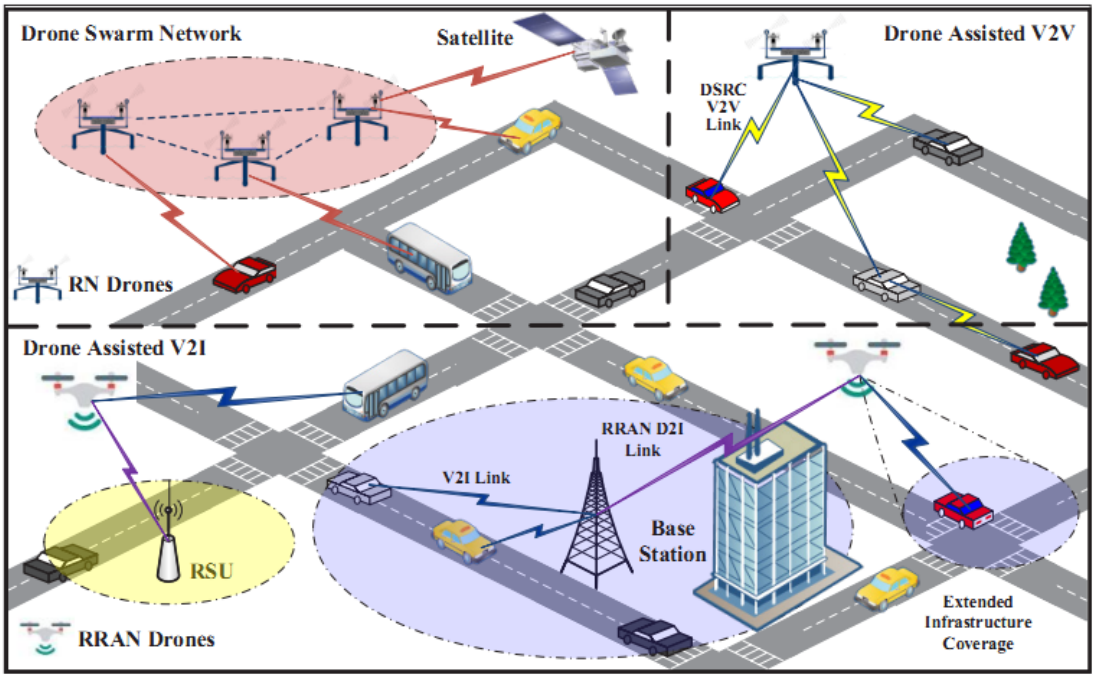

Figure 3.1: Drone-Assisted Vehicular Network architecture [2]

3.2.2 Characteristics of DAVN

Drone-Assisted Vehicular Network presents the following characteristics [2]:

1. **LoS links** Drones flying in the sky have a higher probability to connect ground nodes and other drones via LoS links, which facilitates highly reliable transmissions. Besides, drones can adjust their hovering positions to maintain the quality of links.
2. **Dynamic deployment ability** Drones can also perform as one kind of network infrastructure allowing ground users to access. They can be dynamically deployed/ re-deployed according to real-time requirements, which is more cost-efficient than statically fixed cells.
3. **Drone swarm networks** Benefiting from its high flexibility and rapid provision features, the drone swarm network is a feasible solution to recover communication, especially for

scenarios where communication resources are scarce or unavailable, such as post-disaster environments.

With these characteristics, drones in DAVN can be integrated with classic vehicular network to improve the connectivity between vehicles, extend coverage of infrastructures, facilitate network information collection, and provide additional accessing resources for vehicles [21].

3.2.3 DAVN services

By leveraging drones, DAVN provides a portfolio of messages [2]:

- **Drone assisted safety message broadcast** As flying surveillance nodes, RN drones keep monitoring traffic conditions through embedded cameras or sensors. When emergency issues are detected or reported by vehicles, the corresponding RN drone directly broadcasts safety alerts to all vehicles within its coverage, and its neighbor RN drones in drone swarm networks. By doing so, safety messages can be quickly disseminated over large areas through fewer hops and more reliable LoS links.
- **Drone assisted ubiquitous Internet access** RRAN drones provide infrastructures with mobile extensions to mitigate the dynamic vehicular network topology. The number and positions of RRAN drones can be computed in such a way that their coverage is maximised according to dynamic traffic distribution. Shared by multiple infrastructures, RRAN drones flying among infrastructure coverage gaps can help realize seamless handoff. Not only to complement coverage issue, drones can also be exploited to adjust capacity for certain areas on demand, based on the spatial and temporal traffic dynamics.
- **Drone assisted transportation surveillance** Compared with traditional transportation surveillance systems consisting of fixed cameras, drone assisted transportation surveillance systems are constituted by both vehicles and drones carrying sensors. Hovering over roads, drones can collect network information from their unique perspective. Crowdsourcing algorithms can be employed to process and combine the data gathered by each vehicle or drone, then present accurate traffic and network information to vehicular network controllers.
- **Drone assisted additional spectrum provision** Due to the flexibility and fast deployment features, drone swarm networks are a desirable platform to provide additional spectrum for vehicular network. When licensed vehicular network spectrum is used up in a dedicated area, a drone swarm network is dispatched over it, and builds communications with ubiquitous access points, such as high altitude platforms and satellites. Vehicles leverage-specific communication interfaces running on additional spectrum to access drones, then relay V2V messages, or access ubiquitous access points through drone swarm networks' assistance.

3.2.4 Related work

In the literature, different frameworks of drone assisted vehicular networks are proposed. Authors in [2] define the drone assisted vehicular networks, where the key components, the ad-

vantages and disadvantages are presented. Authors of [22, 23] propose a new framework for using small UAVs as mobile infrastructure nodes in order to enhance the connectivity between vehicles. The UAV-vehicles communication simulator, VEINS (Vehicles in Network Simulation), consisting of the network simulator part and the road traffic model simulator part is presented in [24]. Different models are executed by an event-based network simulator (OMNeT++) while interacting with a road traffic simulator, Simulation of Urban MObility (SUMO) [25]. Authors of [21] tackle the throughput maximization problem with delay constraints in UAV-assisted VANETs to find the best delivery strategy and select the optimal paths for data delivery, with consideration of the links transmission rate and the delay constraints for data dissemination. Authors of [26] propose a software defined space-air-ground integrated network architecture that supports diverse vehicular services in a seamless, efficient, and cost-effective manner.

3.3 Lane Change Assistance State-of-the-Art

Lane change is among the leading causes of motor vehicle accidents. Thus, it has attracted attention of road safety stakeholders. Indeed, lane change is a real-time critical maneuver that requires special treatment from the driver. To be more specific, vehicle driver must pay careful attention to the leading vehicle on their lane and the surrounding vehicles on the target lane, and perform proper actions according to the potential adversarial or cooperative reactions demonstrated by the surrounding vehicles [27]. Therefore, it is of great interest to design and implement an efficient and safe lane change maneuver for vehicular networks.

For reader's clarity, "ego vehicle" is an autonomous vehicle that implements the lane change maneuver, that is to say, the vehicle to be controlled by our designed algorithm. Actually, there is not necessarily only one ego vehicle. For example, in the case of cooperative learning, several ego vehicles will work together and implement the designed algorithm.

Among different lane change maneuvers, DRL based ones are widely studied. The reason is that the lane change scenario is very suitable for DRL, where the agent learns the lane change behavior by try and trials with the reward feedback. To this end, references [28–36] devoted to the design of the DRL model with input feature, action space and reward function.

There are also other techniques for lane change maneuver, for example, formulating the lane change problem as an Optimization Problem (OP) with safety, efficiency and comfort constraints [40]. References [37–43] aim to achieve a safe and smooth lane change by merging different types of information. In the following sections, we will analyse the mentioned papers and provide a comparison table for both lane change decision making using DRL and other techniques.

After study the existing lane change maneuver, we perform a literature study on federated learning in DAVN in order to couple our drone assisted LCA platform with FL. In fact, references [44–58] apply FL to the vehicular network to address challenges such as constrained network coverage, high mobility, and dynamic topology.

In the following sections, we will analyse the mentioned papers in detail.

3.3.1 Lane change decision making using DRL

It is often the case to model the lane change system as a MDP with different state spaces, action spaces and reward functions. In [28], a DQN agent is trained with the Deep Deterministic Policy Gradient (DDPG) algorithm in order to make lane change decisions and avoid collisions. Moreover, authors take into account the update delay of the remote vehicle. Performance results show that the agent learns to make successful lane changes with both lateral and longitudinal control. Nevertheless, the simulation scenario considers a single remote vehicle.

Authors in [30] and [31] train a hierarchical DQN structure to learn both the lane change decision-making (high-level control) and lane change trajectory planning (low-level control). The reward function is defined with the ego vehicle's yaw rate, yaw acceleration and lane changing time in order to learn a smooth and efficient lane change behavior.

In the architecture proposed in [33], the lane change decision is made by maximizing the ego car' s acceleration. The input of the trained Convolutional Neural Network (CNN)-based DQN is a vector of 27 elements, representing the ego vehicle and its 8 neighbors' states. The action space contains 6 elements: stay in the current lane with 4 different accelerations, or change lane to the right or to the left.

Authors in [34] propose a DQN based method using grid-form state representation. The longitudinal speed is controlled by a low-level rule-based trajectory controller. Whereas, the lateral lane change decisions are made by a high-level lateral DQN decision-maker. It is noteworthy that 12 interfering vehicles are considered. However, the research study lacks a collision performance evaluation and a safety guarantee assessment.

Authors in [35] adopt an attention-based DRL to address the interaction with surrounding vehicles while making lane change decision. The inputs are images extracted from the car' s front view. The algorithm performance evaluation under different realistic scenarios reveals that the proposed DRL algorithm is capable of learning both lane change decision and path planning with 73.5% of successful episodes.

In [36], authors considers the overall traffic efficiency instead of the travel efficiency of an individual vehicle. A convolutional DQN is trained with the input consisting of the traffic snapshots.

The following table compares the previous papers according to the following criterion: input feature (input nature, input size, presence of vehicle velocity and position, information about road geometry), reward function (consider maximal speed, collision, lane change smoothness and lane change time), number of surrounding vehicles, model structure, presence of low-level control, lane change trajectory and simulator used.

3.3.2 Lane change decision making using other techniques

Besides DRL, various research studies use different techniques to achieve safe and efficient lane change maneuver. Authors of [37] apply Model Predictive Control (MPC), authors of [38] combine a Kalman filter with rule-based lane change process, while authors of [39] introduce

Table 3.1: Comparison of the papers on lane change decision making using DRL

Reference		[28]	[29]	[30]	[31]	[32]	[33]	[34]	[35]	[36]
Input feature	Input nature	image	image	vector	vector	vector	vector	matrix	image	matrix
	Input size	8	4	6	8	13	27	135	76800	138
	Velocity	√	√	√	√	√	√	√	√	√
	Position/ distance	√	×	√	√	√	√	√	√	√
	Road	×	×	×	√	×	×	×	√	×
Reward function	Maximized speed	√	×	√	×	√	√	√	√	√
	Safety/ collision	√	×	×	×	×	√	√	√	×
	lane change smoothness	×	√	×	×	×	√	√	√	√
	lane change time	√	√	√	×	×	×	×	×	√
Road geometry		Straight	Straight	Straight	Straight	Arbitrary	Straight	Straight	Arbitrary	Straight
Surrounding vehicle number		1	Arbitrary	Arbitrary	Arbitrary	6	8	12	Arbitrary	Arbitrary
NN used		DNN	LSTM	DNN	DNN	DNN	CNN	CNN	CNN	CNN
Low-level control (angle/ acceleration)		×	×	×	×	×	√	×	√	×
lane change trajectory visulization		√	√	√	×	×	×	×	×	√
Simulator		Airsim	highD dataset	Self-built platform	Self-built platform	Unknown	Unknown	Private platform	TORCS	Self-built platform

the noval Graphic Convolution Q-network (GCQ) model. In addition, by integrating different information, such as vehicle states, road constraints, and human demonstrations, the lane change performance is further enhanced.

Reference [40] studies the lane change problem in a scenario mixing Connected Autonomous Vehicles (CAVs) and ordinary vehicles on a highway. The problem is formulated as a Quadratic-Constrained Quadratic Programming (QCQP) problem that takes into account safety, efficiency, and comfort. However, the vehicle state is considered stationary while solving the OP. In [37], a lane change manoeuvre is proposed for lateral and longitudinal control. Firstly, the lane change decision is derived using a series of conditional judgments, based on the presence of obstacles ahead, the relative velocity and distance with surrounding vehicles. Afterwards, the lane change path is generated by an adaptive MPC. According to the simulation results, the mentioned scheme achieves a satisfactory real-time performance.

Authors of [41] propose an offline risk sensitive control by introducing a subjective risk perception module to the lane change maneuver. In this way, the lane change behaviors are personalised for different drivers with different driving styles. In [38], the authors propose a risk assessment prediction for lane change. Specifically, vehicle state is predicted by a Kalman filter. In additions, the vehicle's cooperativeness is estimated from the relative deceleration and the predicted distance between objective vehicle and ego vehicle. As a result, the lane change time is reduced.

Reference [39] considers the lane change decision-making for ordinary vehicles as a multi-agent MDP. The authors train a centralised sharing model based on GCQ for all CAVs in order to maximise the system' s total reward by adopting graph representation of the input feature. The output of the model is an action set for all CAVs. The lane change agent proposed in [42] involves human demonstration to train a Double Dueling DQN (D3QN) in order to make the lane change decision more accurate. The proposed agent is tested on a highway with a ramp where ego vehicle performs lane changes to go off the ramp. Simulation results show that the

proposed strategy achieves good safety performance, and exhibits a good target achievement ability.

In [43], an Observation Adversarial Reinforcement Learning (OARL) approach for robust lane change decision making is introduced. The authors present the lane change behaviors as a constrained observation-robust MDP. Meanwhile, a black-box attack based on Bayesian optimization is modeled and implemented in order to simulate the natural observation uncertainties from sensing and perception system. The experiment results demonstrate that the proposed scheme can make lane change decisions robustly under observation uncertainties.

The previous mentioned papers are summarized in Table. 3.2. With this analysis, we propose our LCA platform coordinated with drones in the next chapter. The proposed platform make lane change decisions based not only on ego vehicle's local information, but also road's global information provided by drones. In addition, the drones hover over the highway will also perform global lane change control to enhance road safety. More details will be discussed in chapter 4.2.

Table 3.2: Comparison of the papers on lane change decision making using other techniques

Reference	**Mtehod**	**Simulator**	**Observations**
[37]	Adaptive MPC	Carsim-Simulink	Real-time performance
[38]	Kalman filter and rule-based lane change process	Carsim-Simulink	Risk assement by cooperation intention
[39]	Graphic Convolution Q-network (GCQ)	SUMO	The output is an action set for all CAVs
[40]	QCQP problem with safety, efficiency & comfort constraints	MATLAB	The vehicle state is considered stationary while solving the OP
[41]	Risk Sensitive Control & Subjective Risk Perception	Real drivers CARLA	Personalizaed driving style; vehicle motion controlled by the expanded bicycle model
[42]	D3QN human demonstrations	CARLA	Tested in an off-ramp scenario
[43]	OARL	SUMO	Black-box attack with Bayesian optimization to find the optimal adversarial observation perturbations

3.3.3 Our research direction

With the analysis in the previous two sections, we may draw the following conclusions:

1. The previous cited papers manage the lane change based on local information related to instantaneous speeds, accelerations and distances. In other words, the literature studies disregard the global traffic state, i.e. the road vehicular density which highly impacts the lane change decision-making.

2. The reward function is static and not dynamically adapted to the performance results and collision ratios.

3. The research studies are established with rule-based models that perform well under predefined operating conditions. These works, however, can fail with presence of risky roads and emergency vehicles, which requires special consideration in unexpected situations.

Thus, we propose our GL-DEAR platform which addresses the aforementioned issues by cooperating with drones, which provide global traffic information to the lane change agent and dynamically adjust lane change agent's reward function. In addition, the drones further enhance the road safety by performing global lane change control, where the corresponding drone will send an ULCR to the vehicle. The details of GL-DEAR platform is presented in chapter 4.

3.4 Federated Learning for Drone Assisted Lane Change Maneuver

3.4.1 Federated learning related work

Federated machine learning has attracted various research studies. The main problems for applying FL to vehicular networks are the communication cost and the model accuracy. Authors in [44–48] strive at achieving reliable federated learning with low communication latency. Papers [45, 48–50] focus on avoiding inefficient local updates that reduce global model's accuracy. Authors in [50–53] apply block-chain method for the security issue in communications. Papers [54–58] tackle the optimization of the energy consumption in vehicular networks.

In [45], the concept of reputation is introduced as a metric to prevent unreliable clients from performing the data poisoning attack, or unintentionally low-quality data due to energy constraints or high-quality data. A reliable worker selection scheme is proposed for federated learning tasks based on this metric. Reference [46] proposes the Communication-Mitigated Federated Learning (CMFL) algorithm, which excludes irrelevant client-side updates, in order to reduce the communication overhead and guarantee learning convergence. It provides clients with feedback on the global trend of model updating. Each client checks to ensure that their update aligns with this global trend and is relevant enough to model improvement. The simulation result confirms that CMFL results in a significantly smaller network footprint in comparison to the most advanced solutions, such as vanilla FL, Gaia, and Federated Multi-Task Learning. In [47], the authors propose a FedVANET algorithm that incorporates a recursive inner-cluster federated learning and an inter-cluster update algorithm to reduce communication load, enhance model performance, accelerate the convergence of the model, and enhance its robustness. Authors of [48] propose a customized partial flexible federated learning algorithm for vehicular edge computing. Only a part of the clients are allowed to participate in the updating based on their local

dataset size. Especially, the central server limits the local upload time in order to aggregate the asynchronous updates at the same time. The proposed algorithm improves the federated algorithm by reducing the communication costs and updating in vehicular edge computing. The authors of [49] propose a selective model aggregation approach based on image quality and computation capability. A greedy algorithm is used to solve the two-dimensional image-computation-reward contract-theoretic problem. In [52], the authors propose an asynchronous advantage actor-critic-based asynchronous federated learning algorithm. This algorithm selects a subset of devices, designs UAVs locations, manages subchannels, and transmits power resources in order to minimize the FL model execution time and learning accuracy loss.

Reference [58] proposes a joint UAV-coalition and Drone-to-Vehicle (D2V)-cell-distribution framework. The UAVs serve as a relay node between the vehicles and the FL server. The introduction of UAVs enhances communication efficiency by reducing communication links and node failures, while simultaneously safeguarding user data privacy as data is still distributed locally on local clients. In [59], the knowledge distillation and dynamic weight adjustment methods are proposed. Knowledge distillation addresses the convergence issue arising from non-identical distributed data, whereas dynamic weight adjustment addresses the performance decay issue arising from imbalanced datasets. Furthermore, the clients have different model structures, and the global model updates only the common part of the model. A blockchain-based decentralized federated learning architecture is presented in [60]. To safeguard data training and contribution verification among UAVs, user privacy protection functions are combined. The scenario comprises numerous UAVs, task publishers, MEC nodes, base stations, and a consortium blockchain. Specifically, the decentralized network is restricted to only authorized nodes registered with the certification authority. The optimal strategies of both task publisher and worker UAVs in the dynamic environment are obtained through a reinforcement learning-based algorithm, albeit without a comprehensive understanding of precise network parameters.

3.4.2 Federated reinforcement learning related work

In order to cooperate FL with DRL-based lane change maneuver, it is necessary to review related work on FRL. In our scenario, the lane change agents will be the local clients to learning from a same environment using their own data, while the drone is the central server to aggregate a global model. Moreover, the local models and the global model are DQNs with the same structure.

Indeed, the combination of deep reinforcement learning with federated learning yields federated deep reinforcement learning as a new computing paradigm, where each user trains a local DQN and the center trains a general DQN. In each learning iteration, the center sends a policy to each user. By following this policy, each user performs an action and receives a reward. Each user then updates her local DQN using the received reward. Also, each user sends her reward signal back to the center, which updates the general DQN based on these reward signals [61].

In the literature, federated reinforcement learning is widely applied for optimal resource allocation in edge computing [61–64]. Another application, that is, the huge task processing delay in edge computing, is tackled [63, 65, 68].

Reference [61] addresses the trade-off between the optimal resource allocation strategy and privacy for Internet-of-Things (IoT) edge computing. The authors propose the CFRL-based resource allocation framework, where each edge host develops a local resource allocation strategy and shares it with the server. The server formulates a resource division strategy and reserve resources for each edge host accordingly. Reference [62] considers the situation where bandwidth of V2I and V2V link and the total amount of edge cloud caches are limited. Then, the bandwidth and cache joint allocation strategy to minimize the weighted average delay of data acquisition is studied. An edge cooperative cache algorithm based on DDPG is further developed. In [63], authors propose a vehicle computing network architecture based on cloud-edge-end collaborative computing, in which cloud servers, edge servers, service vehicles, and task vehicles themselves can provide computing services. The computational offloading strategy is found by the M-TSA algorithm. In addition, task prioritization and computational offloading node prediction are also considered.

In [64], the authors tackle the task processing efficiency; delay-sensitive and computation-intensive tasks by the joint optimization of Computation Offloading and Resource Allocation (CORA) with the objective of minimizing the system cost of processing tasks subject to the processing delay and transmission rate constraints

In order to meet the computation tasks execution delay constraint, reference [65] proposes an optimal task partition ratio of three tasks: the local execution, V2V offloading, and multi-vehicle collaboration. Then, a Multi-Vehicle Intelligent Collaborative Computing strategy (MV-ICCS) is devised to minimize the total system delay. On the other hand, reference [66] formulates the multi task allocation problem as a game, where the goal is to achieve higher allocation utility. To be more specific, the utility consists of task latency, overhead, task transmission model, task priority and edge node' s capacity.

Reference [67] introduces the Attention Transformer to more effectively address the fusion of local updates, this approach exhibits significant improvements compared to FedAvg and non-federated Soft Actor-Critic single-agent methods, including higher episodic rewards. Furthermore, as the number of participating agents increases, the proposed model in this paper demonstrates greater efficiency in the Meta-World environment, a benefit not observed in traditional federated learning methods.

Paper [68] proposes the Federated Deep Reinforcement Learning (FDRL) framework, known as FADE, as a solution to leverage the latency concern due to the dynamic nature of IoT networks. The proposed approach enables the system to autonomously adapt to changing network conditions, improving caching efficiency, which proves the potential of applying FRL for optimizing edge caching in decentralized IoT environments, for improved content delivery and reduced latency.

Authors of [69] claim that existing literature exhibits limitations such as high energy consumption, communication costs, and latency, particularly when dealing with substantial data uploads to computational servers. To resolve these issues, the authors propose an innovative solution that integrates UAVs and introduces a Multiagent Federated Reinforcement Learning (MAFRL)-based resource allocation framework. This approach formulates the connectivity

problem as an optimization challenge, transforming it into a MDP and solving it through a Multi-Agent Reinforcement Learning (MARL)-based resource allocation algorithm. Importantly, their approach incorporates a FL model within the context of Multi-UAV-enabled IoMT. The proposed solution's effectiveness is evaluated through simulations using the heartbeat dataset, offering promise for enhancing healthcare systems and patient safety in challenging network environments.

3.4.3 Our research direction

The majority of the previous mentioned papers pertain to a singular application, namely data dissemination. Moreover, they tackle the federated machine learning without taking into consideration the calibration of the global update frequency. Another major drawback is the delay processing assessment at the drone level.

Contrarily, our work brings the focus to three use cases [70]:

1. Lane change assistance: where the lane change maneuver makes efficient and safe lane change decisions for the ego vehicle in order to surpass a slow vehicle in front or continue the trip.
2. Emergency vehicles: where the leading vehicles should change lane to give way to the emergency vehicle.
3. Random road risk prevention: where the ego vehicle should change lane to prevent driving on the road with potential risks.

When considering real-time lane change assistance use case, decisions should be performed in a tight time window. The emergency vehicle's use case induces deployment of vehicle processes that facilitate the passage of an emergency vehicle and reduces its blocking time. Risk prevention use case prevents the vehicle from driving in the proximity of a random road risk. The three cases induce timely dissemination of time-critical safety signals.

Furthermore, we address the time stringent requirement of road safety services by integrating UAVs' assistance, i.e. in the context of UAV assisted vehicular networks. In fact, UAV assisted vehicular networks combine the benefits of drones and vehicular networks to deal with challenges such as constrained network coverage, high mobility, and dynamic topology. As a result, two important concerns are raised. Firstly, drone's energy consumption is a crucial issue that should be optimized efficiently. More specifically, drone battery life is limited (e.g. 40 minutes [71]) and should be considered while processing vehicular data. Secondly, the fine tuning of the global update frequency is essential to the FL.

We propose an dynamic update frequency adaptation framework which learns to achieve the best trade-off between road active safety provisioning and drone energy consumption. By investigating road safety performance (i.e. collision rate, the risky and impolite driving time on the road) and drone energy consumption with different global update frequencies, we are able to define the thresholds of the update frequency. The details of the proposed DAFL platform are presented in chapter 5.

3.5 Conclusion

In this chapter, we exhibit a literature review about lane change in DAVN. First of all, we introduce the DAVN architecture, the characteristics of DAVN and DAVN services, as well as related work in DAVN. Then, we conduct a comprehensive state-of-the-art study of LCA maneuvers and provide a detailed analysis of the papers. To be more specific, we compare LCA maneuvers using DRL and other techniques. In addition, we study the FRL for drone assisted lane change maneuver.

Based on these understandings, we found it necessary to propose a LCA maneuver in DAVN, where drones can provide global information and thus enhance the LCA. As a result, we devise the GL-DEAR platform, which will be discussed in the next chapter.

Moreover, when applying FL, it is important to consider the global update frequency, which is essential to the trade-off between drone energy consumption and road safety performance. This is achieved by our proposed DAFL framework. The aim is to dynamically adjust the global update frequency to find the best trade-off between drone energy consumption and road safety performance. The details of DAFL framework will be detailed in chapter 5.

Chapter 4
Proposed GL-DEAR Platform

4.1 Introduction

Based on the comprehensive literature study in chapter 3, we propose our LCA platform using DRL approach, called GL-DEAR, which is a joint global and local drone-assisted lane change platform based on deep-Q network with a dynamic reward function.

In order to achieve a satisfactory lane change performance, the reward function is designed from safety, comfort and efficiency perspectives. In particular, the weights of the three rewards are adjusted according to the surrounding traffic condition.

The drones hovering over the highway provide global information (i.e. road vehicular density) to the ego vehicle and perform global control by 1) computing and sending a dynamic collision reward to the ego vehicle; 2) sending an ULCR to the ego vehicle when a road risk exists ahead, or an emergency vehicle behind the ego vehicle is detected.

The proposed lane change platform is tested with the authentic NGSIM dataset. Simulation results prove that the platform is able to perform safe and efficient lane change on a road prone to risks and emergency vehicles.

In particular, our study involves risky lanes and emergency vehicles that require a higher priority level than other vehicles. Consequently, the ego vehicle can acquire an intelligent lane change behavior even under unexpected scenarios. Moreover, the proposed LCA platform takes into account the road vehicular density and achieves a dynamic reward function that adapts to the fluctuating collision ratio. It should be pointed out that the dynamic reward function and the vehicular density are computed in real-time by the drones hovering over the highway.

Compared to the existing literature, our contributions are four-fold:

1. We provide timely and accurate acquisition of traffic flow information of the overall highway with drones' assistance.
2. We devise a lane change decision-making module that integrates global control by drones and local control by DEAR RL agent, which guarantees a safe and efficient lane change with the presence of road risks and emergency vehicles.
3. We design two driving modes for the ego vehicle: speed mode and safety mode, depending

on the traffic condition around the ego vehicle with the purpose of reducing total travel time while avoiding collisions [72].

4. We evaluate the performance of the proposed LCA platform, denoted as GL-DEAR with an authentic dataset, NGSIM, generated on a highway in California [4].

The rest of this chapter is organized as follows: In the second section, we shed the light on our GL-DEAR LCA platform along with its three modules: *Road with emergency vehicles and risks*; *Data file acquisition and processing* and *Real time lane change decision-making*. Specifically, in the data processing module, we compare the model using Principal Component Analysis (PCA) for input feature dimension reduction and without using PCA. In the third section, we present our simulation scenario along with numerical results, as well as a detailed performance analysis. Finally, section four concludes the chapter.

4.2 LCA Platform

4.2.1 LCA modules

Our research strives to achieve an efficient and safe lane change maneuver with three main modules that operate in tandem, as illustrated in Fig. 4.1.

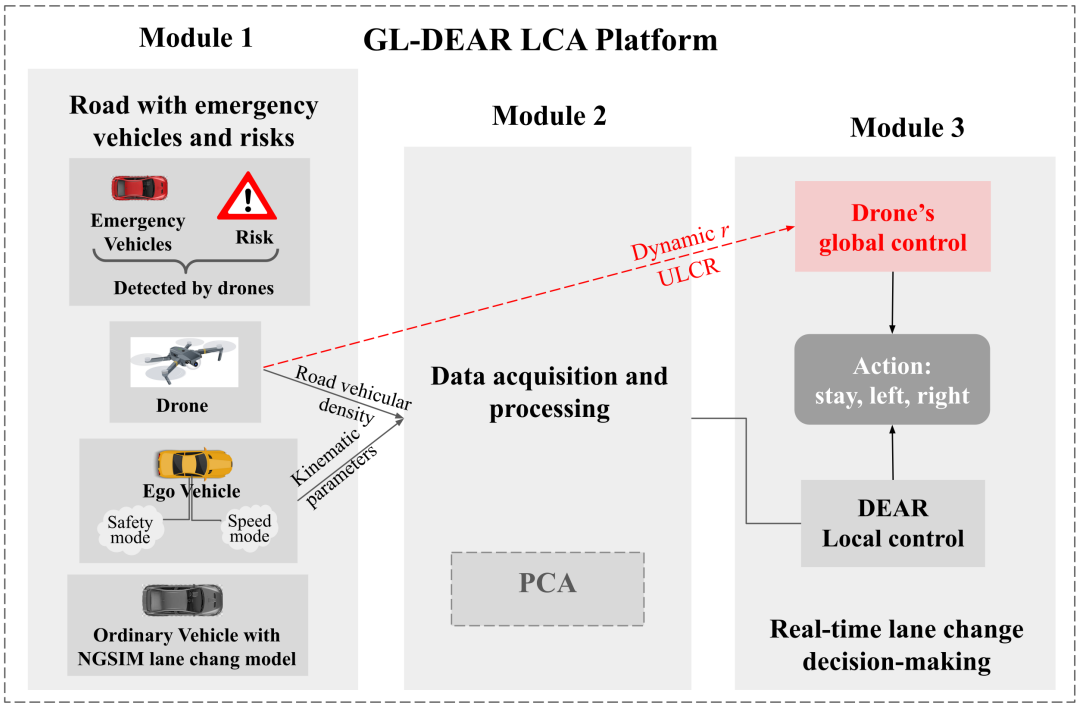

Figure 4.1: Proposed GL-DEAR LCA platform with three modules: Road with emergency vehicles and risks; Data file acquisition and processing and Real time lane change decision-making. The PCA in the dashed box in the data processing module means two data processing methods: one with PCA for input feature dimension reduction, the other without using PCA

Module 1 Road with emergency vehicles and risks

We consider a highway prone to road risks and emergency vehicles. Three kinds of vehicles, namely ego, ordinary, and emergency vehicle move on the highway. Our proposed platform helps the ego vehicle to learn a safe and efficient lane change maneuver, as shown in Fig.4.1.

- It should be noted that ego vehicle adopts two driving modes: speed mode and safety mode. Moreover, it switches between both modes according to its neighbor's number, $n_{neighbor}$:
  - If $n_{neighbor} \leq n_{safe}$ (predefined threshold), enter speed mode.
  - If $n_{neighbor} > n_{safe}$, enter safety mode.
- Emergency vehicles are ambulances and police cars that have a higher priority than ordinary vehicles. More specifically, an ego vehicle should change lane to give way to a following emergency vehicle.
- Ordinary vehicles move according to the NGSIM dataset. In fact, we consider a real world scenario that lies on the NGSIM lane change model, trained from the authentic dataset NGSIM. The training of the NGSIM lane change model is detailed in 4.2.2.

Drones hovering over the highway assist the lane change decision making with the global control by the following steps:

- At the end of every training epoch, the drone updates and sends the dynamic collision reward r, as well as the road vehicular density to the ego vehicle.
- As long as the drone detects a road risk ahead (i.e. construction work or car accident), or an emergency vehicle behind the ego vehicle, it sends an ULCR to the ego vehicle to force it to change lane. After receiving the ULCR from the drone, the ego vehicle will try to change lane as soon as possible during the valid time of the ULCR, t_{ULCQ}.

Module 2 Data file acquisition and processing

In the pre-lane change phase, data is acquired and collected by the ego vehicle. The collected data file stores kinematic parameters, i.e. GPS coordinates, velocity and acceleration that are retrieved by vehicle sensors in addition to the road vehicular density, retrieved by the drones. Afterwards, data files are trimmed in the pre-training phase, as they may contain some irrelevant information for lane change decision.

In this module, we propose two data processing functions: one without data feature dimension reduction, the other with data dimension reduction using PCA and a regression algorithm in order to remove irrelevant and inefficient data. In fact, PCA is a data reduction ML algorithm by finding k vectors on which to project the data while minimizing the projection error [73].

The data reduction process in our work contains following steps:

1. Scale the features using Equation. (4.1). In this way, each feature is scaled and translated

individually such that it is between zero and one, without breaking the sparsity of the dataset [74].

$$x_{std} = (x - x_{min})/(x_{max} - x_{min})$$
$$x_{scaled} = x_{std} \cdot (x_{max} - x_{min}) + x_{min} \tag{4.1}$$

2. Reduce feature dimension using PCA. The reduced dimension depends on the number of principle components, n, that we choose. n is chosen by using the **explained variance** v, a measure that maps the variance in the original data to the low-dimensional model, expressed by the eigenvalues λ, i.e.,

$$v_i = \lambda_i / \sum_{j=1}^{n} \lambda_j$$

Fig. 4.2, 4.3 and 4.4 show the explained variance ratio of each feature accounted for the whole dataset with 90%, 95% and 99% explained variance. One can see that in order to keep 99% of the variance of the dataset, the first 39 components should be chosen; to keep 95% (respectively 90%,) of the explained variance, 28 (resp. 19,) components should be chosen [73]. In the rest of the work, we keep 99% of the explained variance, which leads to a reduced dimension equals 39, in order to achieve the most accurate learning.

Finally, processed data is fed into the real-time lane change decision-making module. It should be noticed that both data processing mehods (using PCA and without using PCA) are evaluated in the simulation results section. More specifically, the performance results without using PCA is presented in section 4.3.2, while the performance results with PCA is presented in section 4.3.3

Module 3 Real time lane change decision-making

This module integrates the drone's global control and DEAR's local control in order to assist the driver to take the real-time lane change decision at the optimal instant of time. The local control based on DEAR agent is described in section 4.2.3.

The overall algorithm of GL-DEAR platform is described in Algorithm 2.

4.2.2 Real-world scenario based on NGSIM dataset

As explained previously, we trained a NGSIM lane change model from the authentic dataset NGSIM for ordinary vehicles. The NGSIM dataset, retrieved by the National Highway Traffic Safety Administration, includes detailed vehicle trajectory data from 4 different neighborhoods [4]. In this chapter, we adopt the trajectories on a US101 highway to train the NGSIM lane change model. These trajectories are spread over a 6-lane highway without crossings, traffic lights and pedestrians; the lane 6 is a ramp with vehicles coming in and out from the highway. Our study investigates the trajectories on lanes 1 to 5 for the purpose of preventing the exit navigation issue.

For readers' clarity, NGSIM lane change model refers to the authentic model trained from the NGSIM dataset. On the other hand, DEAR is the DRL-based lane change agent implemented

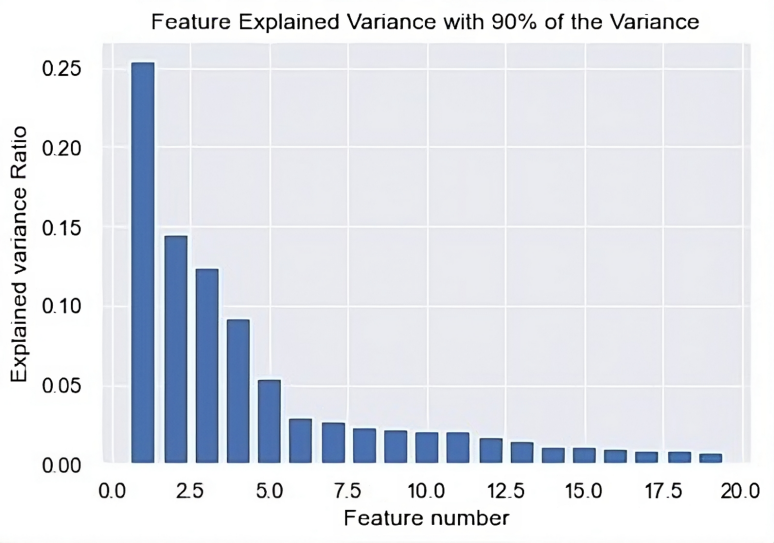

Figure 4.2: Explained variance ratio of the features with 90% explained variance

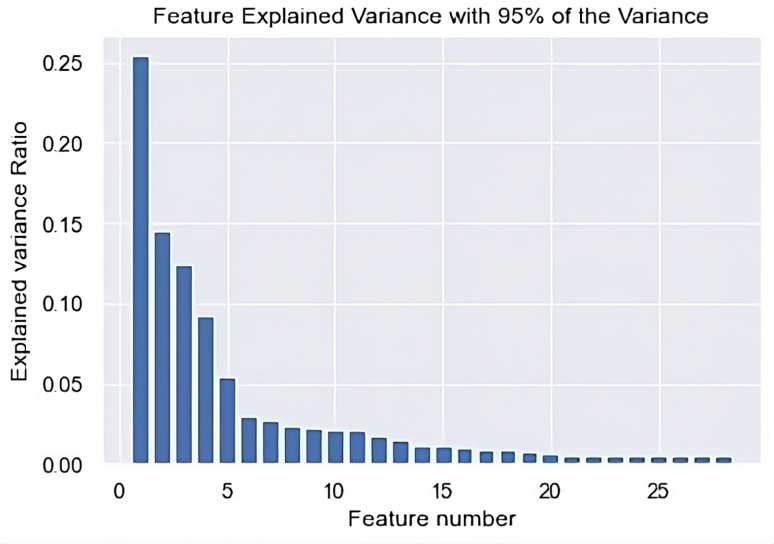

Figure 4.3: Explained variance ratio of the features with 95% explained variance

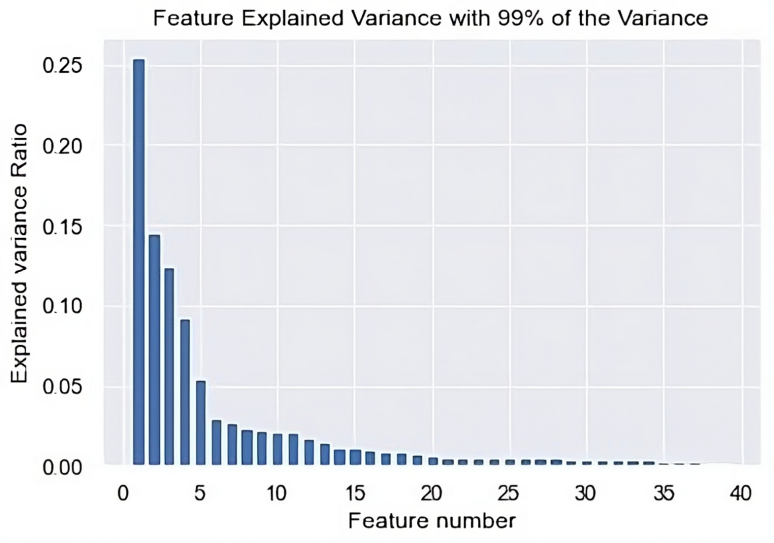

Figure 4.4: Explained variance ratio of the features with 99% explained variance

Algorithm 2 `Algorithm for GL-DEAR platform`

Input: E is the number of training epochs, S is the number of steps per epoch, N is the number of ordinary vehicles on the road, t_{ULCR} is the valid time of the ULCR sent by drone, a_g is the global lane change action, a_l is the local lane change action, and a is the final lane change decision
Initialize $t = 0$, $a_g = 0$
for *epoch* $\leftarrow 0$ to E **do**
    for *step* $\leftarrow 0$ to S **do**
        for $i \leftarrow 0$ to N **do**
            Ordinary vehicle i predicts its lane change action a_i by NGSIM lane change model
        end for
        Ego vehicle computes $n_{neighbor}$
        if $n_{neighbor} \geq n_{safe}$ **then**
            Enter safety mode
        else
            Enter speed mode
        end if
        Ego vehicle predicts a_l by DEAR ▷ Local control
        if risk or emergency vehicle detected by the drone **then** ▷ Global control
            $t = t + 1$
            if $t \leq t_{ULCR}$ **then**
                $a_g = \{1, 2\}$ according to ego vehicle's current lane
            else
                $t = 0$
                $a_g = 0$
            end if
        end if
        if $a_g == a_l$ **then**
            $a = a_g$
        else
            $a = max(a_g, a_l)$
        end if
        Perform lane change for all vehicles
    end for
end for

in the ego vehicle. Both models will be detailed in the following sections. This subsection is dedicated to the lane change model for ordinary vehicles.

Machine learning model applied to NGSIM dataset

In this thesis, we assume that ordinary vehicles adopt the NGSIM lane change model: these vehicles are denoted as NGSIM vehicles in the NGSIM training. In this context, two questions arise: which kinematic parameters in the dataset are most relevant to lane change decision-making, and which model structure should be adopted. Existing literature has addressed these highlighted issues.

In [75], authors take 40 observation frames of each trajectory as the input for the decision stage. Then, detailed data processing is performed in order to achieve better classification accuracy. In [76], the start of a lane change execution process is defined as the time when the vehicle's lateral position starts to change continuously in one direction. Then, the authors adopt the Symmetric Exponential Moving Average algorithm to smooth the peak values during the processing phase. Authors in [77] provide a comprehensive data analysis, based on the velocity differences between the vehicle to change lane and its surrounding vehicles, as well as the gap between them. Afterwards, four lane change models are trained based on the new dataset, namely Logistic Regression, Adaptive Boosting, Extreme Gradient Boosting (XGBoost) and Advanced XGBoost. According to the performance results, the advanced version of the XGBoost model achieved the highest accuracy of 98.9%.

Based on this survey study, we adopt the XGBoost machine learning model that tackles two main features: the surrounding vehicles' velocity and the vehicles' separating distance.

NGSIM dataset features

We extract 9 features from the dataset, $(v_0, v_1, v_2, v_3, v_4, y_1, y_2, y_3, y_4)$, where:

- v_0 is the current velocity of the NGSIM vehicle.
- $v_i; i = 1,2,3,4$ refers to the current velocity of the current lane leading vehicle (respectively the current lane following vehicle, resp. the target lane leading vehicle, resp. the target lane following vehicle).
- $y_i; i = 1,2,3,4$ denotes the distance between the NGSIM vehicle and the 4 neighbors (the leaders and followers in the current and target lanes).

Training and testing for the NGSIM lane change model

We adopt a classifier based on the XGBoost algorithm that is trained with previously obtained training set and labels [77]. XGBoost is an ensemble learning algorithm meaning that it combines the results of many models, called base learners (i.e. Decision Trees) to make a prediction. We apply the grid-search method with the aim of finding the optimal parameters for the XGBoost model. As a result, the optimal classifier achieves the testing accuracy of 98%, which is adopted by ordinary vehicles to produce realistic lane change behaviors.

4.2.3 DEAR for the ego vehicle

This subsection is dedicated to the DQN lane change agent implemented on the ego vehicle.

State space

The agent to be trained is a three-layer DQN with 64, 128, 64 hidden nodes on the first, second, and the third hidden layer. The output is the lane change decision. The state space at time step j consists of 48 kinematic parameters of ego vehicle and its 6 possible neighbors, namely the leader, left leader, right leader, follower, left follower and right leader, as illustrated in Fig. 4.5. The observation can be expressed as

$$o[j] = \{o_{ego}[j], o_1[j], \cdots, o_6[j]\}\text{, where:}$$

- $o_{ego}[j] = \{l_{risk}[j], x_{ego}[j], y_{ego}[j], v_{ego}[j], a_{ego}[j], l_{ego}[j]\}$ is the set of ego vehicle parameters. This set consists of the risk label of current lane l_{risk}, (0 refers to no risk detection, 1 refers to risk detection), horizontal position x, vertical position y, longitudinal speed v, acceleration a, and current lane id l at time j.
- $o_i[j] = \{x_i[j], y_i[j], v_i[j], a_i[j], l_i[j], d_i[j], p_i[j]; i \in [0, 6]\}$ consists of 7 parameters of the i-th neighbor, representing the horizontal position x, vertical position y, longitudinal speed v, acceleration a, current lane id l, distance to the ego vehicle d, and vehicle priority p (0 for normal vehicles and 1 for emergency vehicles such as ambulance and police car) at time j.

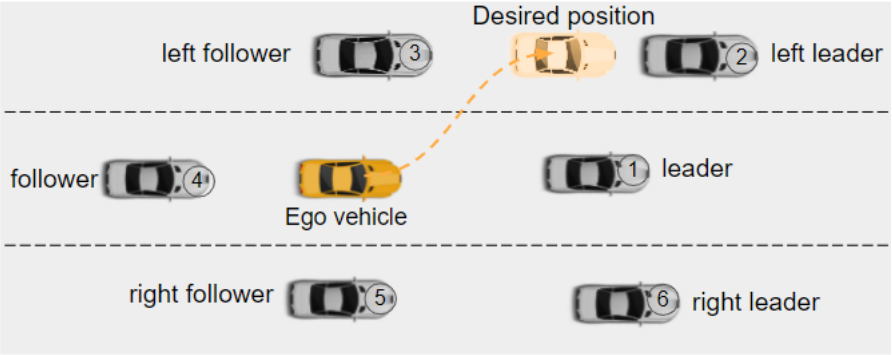

Figure 4.5: Ego vehicle surrounded by six neighbors, trying to change lane to the desired position.

Action space

The action space is defined as $a = \{0, 1, 2\}$ where 0 refers to staying in the current lane, 1 indicates performing a lane change to the right and 2 denotes performing a lane change to the left.

Reward function

The reward function is designed from three different perspectives: comfort (avoid hard brakes and sharp accelerations), efficiency (reduce travel time), and safety (avoid collisions and potential risks) [78]. Consequently, the total reward is the weighted sum of the three rewards:

$$R = w_{comf} R_{comf} + w_{eff} R_{eff} + w_{safe} R_{safe} \tag{4.2}$$

where R_{comf} is the comfort reward, R_{eff} the efficiency reward, and R_{safe} the safety reward, w_{comf} (, resp w_{eff}, w_{safe}) is the coefficient controlling the comfort weight (, resp efficiency weight, safety weight).

- Comfort reward: The comfort reward is negatively correlated to the fluctuation of acceleration, as shown in (4.3).

  $$R_{comf} = -\dot{a}_x^2 \tag{4.3}$$

  $\dot{a}_x$ is the acceleration jitter computed from two adjacent steps.

- Efficiency reward: The efficiency reward consists of two rewards, speed reward R_v and lane change reward R_{change}, as indicated in the following equation.

  $$R_{eff} = R_v + R_{change} \tag{4.4}$$

  The speed reward, R_v is defined as $R_v = -|v_{max} - v|$ such that the closer the speed to the maximum allowed speed is, the higher the reward is. The lane change reward, R_{change} is defined as (4.5), where α is a constant, so as to penalize the lane change agent each time a lane change occurs.

  $$R_{change} = \begin{cases} -\alpha, & \text{if change lane} \\ \alpha, & \text{if stay in lane} \end{cases} \tag{4.5}$$

- Safety reward: The safety reward is the sum of collision reward R_{colli}, vehicular density reward R_{den}, risky reward R_{risk}, and blocking reward R_{block} as follows:

  $$R_{safe} = R_{colli} + R_{den} + R_{risk} + R_{block} \tag{4.6}$$

  The collision reward $R_{colli} = r$ is a dynamic value computed by the drone processor and sent to the ego vehicle periodically. At the end of an epoch, the drone calibrates r according to the following process.

  - Whenever a collision occurs, r is decreased aiming to penalize the collision.
  - In case no collision occurs during a certain time window, r is increased with the aim of encouraging the agent performing lane change.

  The density reward R_{den}, is negatively related to the number of vehicles on the road: $R_{den} = -n_v$, where n_v is the number of vehicles. In fact, the higher the vehicular density is, the higher the risk of collision is at lane change and the lower the reward function is. It is noteworthy that the traffic density is computed by the drone when collecting the *Hello* packets sent by vehicles.

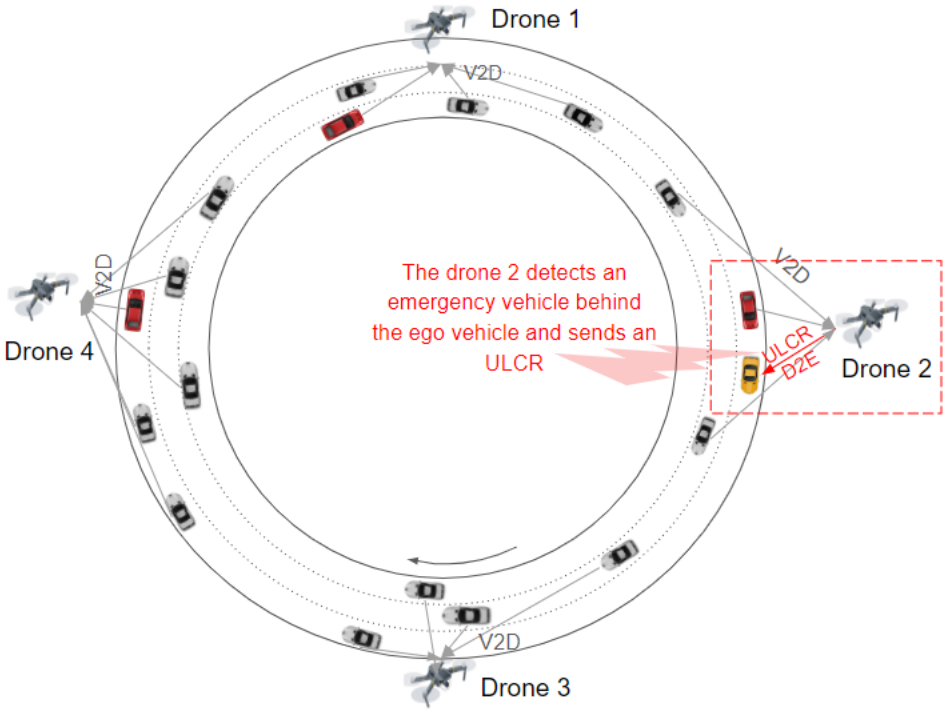

Figure 4.6: 4-km circular highway with V2D and D2E communications. Ego vehicle in yellow, ordinary vehicles in green.

The risky reward R_{risk}, (respectively the blocking reward R_{block}) is inversely associated with the total time of the ego vehicle driving on the risky lane (respectively in front of an emergency vehicle).

4.3 Simulation and Performance Results

4.3.1 Simulation setup

We conduct extensive simulation batches with SUMO [25] in order to test the performance of our proposed LCA platform. The RL agent is implemented by the Gym module in Python [79]. As shown in Fig. 4.6, a 4 km circular highway where ego vehicle tries to perform a collision-less lane change is considered. Ordinary vehicles adopt the NGSIM lane change model, which has been detailed in section 4.2.2. Four drones are positioned above the highway, each with a communication range of 1km [2]. They communicate with vehicles over V2D links and are responsible to calculate the traffic density in their vicinity. Additionally, they compute an adaptive parameter r and disseminate it to the ego vehicle over the Drone-to-Ego (D2E) link.

Vehicle mobility model Vehicles move according to the Krauss mobility model [80] with the maximum velocity of 100 km/h. In order to make the scenario more realistic, a small number of vehicles are set to be aggressive with rude behaviors, such as staying in the leftmost lane for a long time or exceeding the speed limit. Furthermore, some other vehicles are considered as emergency vehicles that force in front vehicles to trigger lane change.

Risk model At the start of each epoch, a random risk, such as a car accident or construction works in front of the ego vehicle is detected. The risk is evenly distributed between the three lanes and occurs one at a time. When a risk is detected, the corresponding lane is deemed "risky" and the ego vehicle should consider changing lanes to avoid driving on the risky lane.

In each simulation step, the agent predicts the next action with ego vehicle's current state retrieved by the Traffic Control Interface (TraCI) of SUMO [93]. Then, the ego vehicle updates its state and computes the reward according to (4.2). The tuple (action, state, reward) is stored in the replay buffer. The details of the performance analysis are provided in the following section.

4.3.2 Performance of GL-DEAR platform without using PCA in Module 2

Baseline models

We compare our GL-DEAR platform to the following baseline models:

- K-Nearest Neighbors (KNN): KNN is a non-parametric supervised machine learning model that uses proximity to classify or predict grouping of individual data points.
- Deep Neural Network (DNN): DNN is an artificial neural network with multiple layers between the input and output layers. It that can model complex non-linear relationships.
- LC2013: LC2013 is the default lane change model in SUMO which discriminates 4 kinds of lane change: (1) strategic lane change to continue the trip, (2) cooperative lane change to allow others to change, (3) speed gain lane change which allows for faster speed, and (4) obligation to drive on the right for emergency such as when an ambulance or police car pass by [25].
- Policy Gradient (PG): PG is a reinforcement learning algorithm. It increases or decreases the probability of taking an action based on the reward obtained. The reward function is described in (4.2).
- DEAR: It is a version of GL-DEAR that does not incorporate global control nor the two driving modes [72].

It is to be noted that the KNN model adopts the default structure by scikit-learn, and the DNN model consists of three hidden layers of size 64, 128, 64 [74]. In addition, the two models are trained with a dataset generated with the same settings as GL-DEAR. As a result, the testing accuracy of the KNN and DNN classifier is 94.2%, 93.1%.

Performance analysis

We compute several performance parameters, namely: collision number, average speed, number of lane change requests (LC requests), time spent in risky lanes (Risky time, t_r), and time spent in front of an emergency vehicle and thus impairing its passage (Blocking time, t_b) during simulation. Table 4.1, 4.2 and 4.3 show the performance of the 5 models tested with sparse (50

vehicles), medium (150 vehicles) and dense (250 vehicles) traffic density. Detailed analysis will be provided in the following paragraphs.

Collision number The primary objective for this research is to reduce fatalities caused by accident-related lane changes, which makes the collision number the most critical performance parameter. According to Tables 4.1, 4.2 and 4.3, all of the models succeed to avoid collisions in sparse, medium, and dense traffic scenarios. But one can see that KNN, DNN and LC2013 induce much lower average lane change numbers than PG, GL-DEAR and DEAR in sparse and medium scenarios. Given this, we found it important to investigate the number of LC requests.

Number of lane change requests The number of LC requests indicates how differently machine learning and reinforcement learning agents behave. The fact that KNN and DNN achieve fewer LC requests than PG, GL-DEAR and DEAR implies that KNN and DNN are less likely to allow lane changes, even when there is a risk or an emergency vehicle around. On the contrary, PG and GL-DEAR reinforcement learning agents attempt to achieve higher reward by interacting with the risky lanes, ambulances, and surrounding vehicles; which leads the agents to take more lane changes.

One can see that the LC requests number of LC2013 has increased dramatically in the dense scenario. This is expected because the LC2013 model implements cooperative lane change, which forces the ego vehicle to change lane and give way to other vehicles, thus larger LC requests number in dense traffic.

Average speed One of the reasons for lane change is overtaking a slow vehicle in front so as to obtain higher speed and reduce the total travel time. This leads us to compute the average speed of the ego vehicle during each test episode. As it can be noticed from Tables 4.1, 4.2, and 4.3, GL-DEAR achieves higher average speed than KNN and DNN in sparse, medium, and dense scenarios. When compared with DEAR, LC2013 and PG, although the average speed of GL-DEAR is lower than the three models, the performance from the safe perspective, i.e. the risky time and blocking time is greatly improved. The reason is that, in GL-DEAR, the frequent change between safety mode and speed mode tunes the trade-off between safety and travel efficiency. The agent learns to increase the speed under the premise of ensuring safety. It should be pointed out that the average speed of GL-DEAR has a decrease of 9.8%, 26.6% and 24% when compared to DEAR in sparse, medium and dense traffic scenarios, while the risky time and blocking time are reduced by 16.1%, 27.1%; 29.4%, 28.4%; and 59.3%, 28.8% for the three scenarios.

Risky time (t_r) We introduce risky time in order to evaluate the agent's performance with road prone to risks. According to Tables 4.1, 4.2, and 4.3, GL-DEAR achieves the shortest risky time compared to the other models in sparse, medium, and dense traffic. Indeed, our model takes into consideration the risky time in the reward function. Consequently, the agent will adapt its behavior in the presence of a risky lane and thus forces the driver to change the lane near an accident or construction works. However, even trained with the same reward function, PG agent has much longer risky time than GL-DEAR while achieving higher average speed.

The reason is that GL-DEAR changes between the two driving modes, trying to optimise the trade-off between safety and efficiency. However, PG agent decides to make more effort to gain higher average speed to increase the total reward.

Blocking time (t_b) The time spent in front of an emergency vehicle is correlated with the driver's cooperation willingness. The higher the cooperation willingness is, the sooner the driver will change lane to give way. As pointed out by Tables 4.1, 4.2, and 4.3, GL-DEAR outperforms KNN, LC2013 and DEAR in the three scenarios owing to R_{block} included in the reward function. Consequently, the agent learns to adjust its behavior when confronted with ambulances, and thus gives way to emergency vehicles, regardless of traffic density. Since KNN and DNN tends to stay in one lane during the whole trajectory, the blocking time will increase dramatically when the traffic density increases, as there will be more emergency vehicles behind the ego vehicle.

Table 4.1: Models performance tested with sparse traffic

Model	**KNN**	**DNN**	**LC2013**	**PG**	**DEAR**	**GL-DEAR**
Collision Number	0	0	0	0	0	0
LC Request	0.6	23	31.5	544	353.3	477
Avg Speed (km/h)	29.9	46.3	60.9	60.8	65	58.6
t_r **(s)**	26.1	22.8	14.4	19	5.6	4.7
t_b **(s)**	83.3	22.8	66.4	100.4	68.7	50.1

Table 4.2: Models performance tested with medium traffic

Model	**KNN**	**DNN**	**LC2013**	**PG**	**DEAR**	**GL-DEAR**
Collision Number	0	0	0	0	0	0
LC Request	69	34.7	44.2	782.5	284.3	516
Avg Speed (km/h)	33.8	47.3	60	61.6	67.1	53
t_r **(s)**	29.1	29.6	25.1	46	10.9	7.7
t_b **(s)**	131.6	66	166.8	172	116	83

Table 4.3: Models performance tested with dense traffic

Model	**KNN**	**DNN**	**LC2013**	**PG**	**DEAR**	**GL-DEAR**
Collision Number	0	0	0	0	0	0
LC Request	44.3	65.5	202.2	717	296.8	466
Avg Speed (km/h)	44	47.1	56.5	55.4	65	49.4
t_r **(s)**	46.2	38.7	20.7	26.4	19.9	8.1
t_b **(s)**	248.2	178.3	82	315.6	106.5	75.8

Impact of simulation parameters

In this section, we compare the performance with different parameter settings, namely n_{safe}, w_{comf}, w_{eff} and w_{safe}. Fig. 4.7 shows the evaluation of w_{safe} when ego vehicle changes between the safety mode and the speed mode. In the simulation, w_{comf}, w_{eff}, and w_{safe} are set to 1,

1, 3 in the safety mode, and 1, 1, 1 in the speed mode. It can be observed that, with larger n_{safe}, w_{safe} less becomes 3. On the other hand, with higher traffic density, w_{safe} becomes 3 more often.

As a matter of fact, these three weights are tuned by n_{safe} so as to recognize the trade off between the travel efficiency and safety. A higher w_{safe} than w_{comf} and w_{eff} means higher penalties for collisions. Consequently, the model learns to adopt very conservative behavior even at the risk of being unable to achieve the desired speed. On the other hand, When w_{comf} and w_{eff} are tuned as the same importance as w_{safe}, all of the three rewards dominate. This setting will encourage the vehicle to make lane changes to reach destination as fast as possible while considering safety [39].

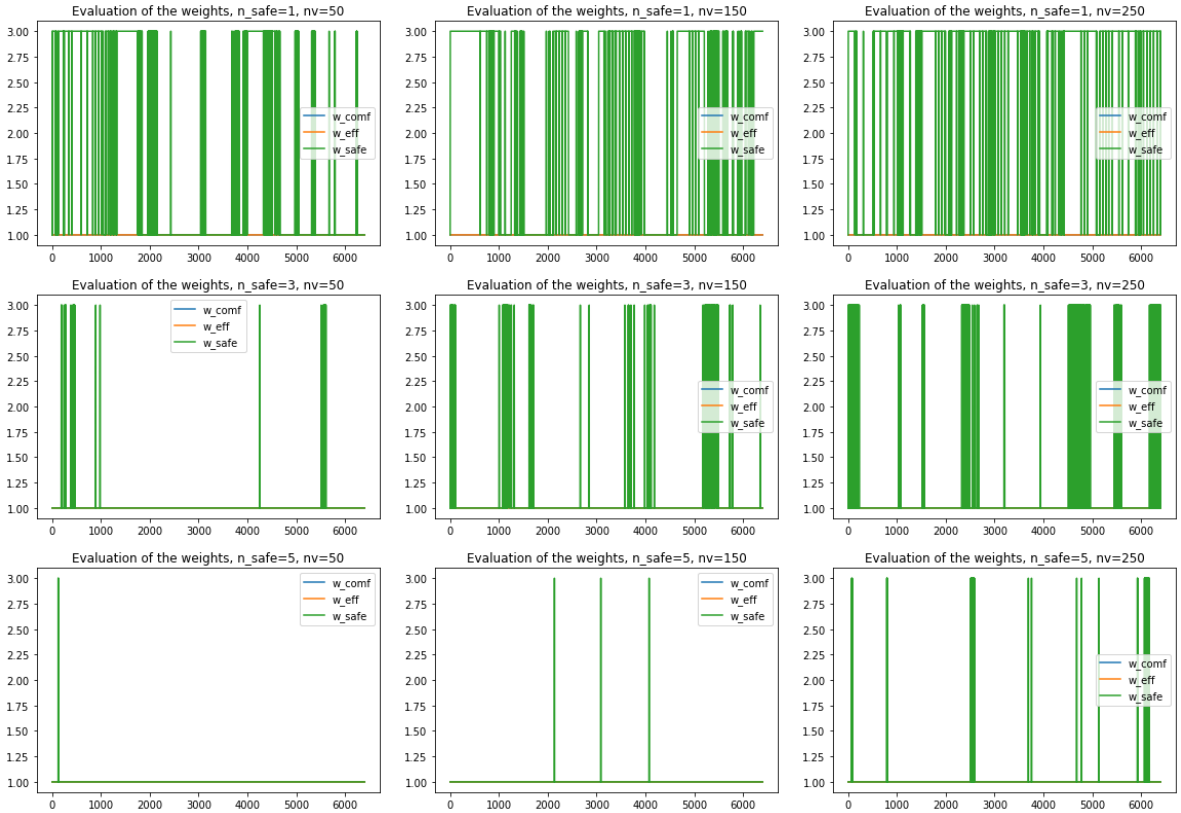

Figure 4.7: Evaluation of w_{safe} with different values of n_{safe}

Table. 4.4 shows the performance of GL-DEAR platform with different n_{safe} in different traffic densities. As one can find out that the best trade-off between travel efficiency and safety is achieved when $n_{safe} = 3$.

Table 4.4: Performance of GL-DEAR with different n_{safe}

Vehicle Density	**Sparse**			**Medium**			**Dense**		
n_{safe}	1	3	5	1	3	5	1	3	5
Avg Speed (km/h)	56	58.6	60.7	52.5	53	55.3	45.9	49.4	49.6
t_r **(s)**	5.2	4.7	18.5	41.7	7.7	13.8	5.6	8.1	9.2
t_b **(s)**	47.3	50.1	64.4	65.6	83	90.9	94.4	75.8	85.9

4.3.3 Performance of GL-DEAR platform with PCA in Module 2

Baseline models

Baseline models are the same as described in section 4.3.2 with Logistic Regression (LR), a supervised machine learning algorithm. It uses a logistic function to model the dependent variables. The LR model is also built with the scikit-learn package and the classification accuracy is 96%.

Performance analysis

We compute several performance parameters, namely: collision number, average speed, number of lane change requests (LC requests), time driving in risky lanes (Risky time, t_r), and time spent in front of an emergency vehicle (Blocking time, t_b) during simulation. Table 4.5, 4.6 and 4.7 show the performance of the 6 models tested with sparse (50 vehicles), medium (150 vehicles) and dense (250 vehicles) traffic densities [72].

Collision number The primary objective for this research is to reduce fatalities caused by accident-related lane changes, which makes the collision number the most important performance parameter. According to Tables. 4.5, 4.6 and 4.7, GL-DEAR, PG, KNN and DNN succeed to avoid collisions in sparse, medium, and dense traffic scenarios. But one can see that KNN and DNN induce much lower average lane change numbers than GL-DEAR in all of the three scenarios. It should be pointed out that a collision-less agent is solely important when lane changes occur. Given this, we found it crucial to investigate the number of LC requests.

Number of lane change requests As a matter of fact, the number of LC requests indicates how differently machine learning and reinforcement learning agents behave. The fact that KNN, DNN and LR agents achieve fewer LC requests than GL-DEAR, DEAR and PG implies that KNN, DNN and LR agents are less likely to allow lane changes, even when there is a risky lane or an emergency vehicle behind. On the contrary, GL-DEAR reinforcement learning agent attempts to learn a safe and efficient lane change maneuver by interacting with the complex environment with its surrounding neighbors, risky lanes and emergency vehicles; which leads to a high number of lane changes.

Average speed The most common reason for changing lanes is to overtake a slower vehicle in order to increase speed and reduce total travel time. Thus, we compute the average speed of the ego vehicle during each testing simulation. As it can be noticed from Tables. 4.5, 4.6, and 4.7, GL-DEAR achieves higher average speed than KNN and LR in sparse, and medium scenarios. When compared with DEAR and PG, although the average speed of GL-DEAR is lower than the two models, the performance from the safe perspective, i.e. the risky time and blocking time is greatly improved. The reason is that the frequent change between safety mode and speed mode tunes the trade-off between safety and travel efficiency. The agent learns to increase the speed under the premise of ensuring safety.

Risky time (t_r) We introduce risky time in order to evaluate the agent's performance with roads prone to risks. According to Tables. 4.5, 4.6, and 4.7, GL-DEAR achieves the shortest risky time compared to the other models in all of the three traffic scenarios. In fact, the reward function of GL-DEAR includes a reward related to the risky time. Consequently, the agent will learn to perform timely lane change in the presence of potential road risks.

Blocking time (t_b) As a matter of fact, the blocking time reflects the driver's cooperation willingness with other vehicles, specifically emergency vehicles. A driver with high cooperation willingness will give way at once when an urgent lane change demand from the following vehicles arises. As shown in Tables. 4.5, 4.6, and 4.7, GL-DEAR outperforms DEAR, PG, KNN and DNN in the three scenarios owing to R_{block} included in the reward function. As a result, the agent can adapt its behavior and gives way when confronted with emergency vehicles, regardless of traffic density.

Table 4.5: Models performance tested with sparse traffic

Model	KNN	DNN	LR	PG	DEAR	GL-DEAR
Collision Number	0	0	2	0	0	0
LC Request	2.25	0.8	5	581.3	75	604
Avg Speed (km/h)	56.6	64.9	48.6	61.3	76	57.7
t_r **(s)**	82.6	34.9	74.5	19.1	46	13
t_b **(s)**	248.6	768.3	0.4	191.9	267.8	91.2

Table 4.6: Models performance tested with medium traffic

Model	KNN	DNN	LR	PG	DEAR	GL-DEAR
Collision Number	0	0	4	0	3	0
LC Request	0.6	1	7	858.6	88	534.5
Avg Speed (km/h)	48.7	57.6	46	61.1	67.9	54
t_r **(s)**	51.5	85.4	44.3	87.3	11.1	8.3
t_b **(s)**	483.4	895.5	71.1	332.1	569.2	83.2

Table 4.7: Models performance tested with dense traffic

Model	KNN	DNN	LR	PG	DEAR	GL-DEAR
Collision Number	0	4	8	0	8	0
LC Request	1.5	22	67	749	158.8	641.6
Avg Speed (km/h)	50.5	54	42.8	55.9	61.1	49.3
t_r **(s)**	44	59.5	116	13.2	74.3	7.12
t_b **(s)**	620.4	670	5.6	443.4	492	229.5

Compared with GL-DEAR without using PCA

With previous performance results, we found that compared with using PCA technique, the GL-DEAR performs better without using PCA. In fact, dimension reduction is applied when

the original high-dimensional space is filled with redundant and noisy information, leading to errors in practical applications and hindering accuracy. In this case, dimension reduction extracts the core data structure, reducing errors stemming from redundancy and noise. This not only enhances accuracy but also simplifies calculations and aids visualization, while fundamentally separating valuable information from the superfluous. However, in our case, the input consists of vehicle kinematic parameters collected by the ego vehicle, which is less redundant and noised. Consequently, applying PCA leads to losing informative training data and thus reduced performance.

4.3.4 Conclusion for the performance analysis

At this stage, we may draw the following conclusions for the GL-DEAR platform:

- According to the previous analysis, the drone-assisted GL-DEAR platform achieves satisfactory performance under testing scenarios with the authentic NGSIM dataset. Indeed, the vehicle density computed by drones highly impacts the reward function, assigning a negative penalty for sparse, medium, and heavy traffic distributions. This fact is demonstrated by the higher performance of GL-DEAR compared to other models. GL-DEAR successfully avoids collisions while taking into account the perspectives of safety, comfort, and efficiency.
- When compared to PG and DEAR, the integration of the global control and the two driving modes achieves huge improvement in the safety perspective.
- Despite the presence of potential risks and emergency vehicles, GL-DEAR agent can learn and adapt to the complex environment and achieve collision-less lane changes.

4.4 Conclusion

In this chapter, we present our proposed GL-DEAR, a drone-assisted collision-less LCA platform based on DRL approach. Specifically, GL-DEAR incorporates drones to collect and process vehicular traffic data. Further, GL-DEAR assists the driver in taking the lane change decision at the optimal instant of time with a comprehensive reward function that takes into consideration road safety, travel efficiency, and passenger's comfort. The performance is further enhanced by the global control of the drones and the two driving modes possessed by ego vehicle. In addition, we evaluate GL-DEAR' s performance in a real-world scenario with an authentic dataset, NGSIM. The simulation results prove that GL-DEAR successfully achieve collision-less trips on a highway prone to risks and emergency vehicles.

In the next chapter, we apply FL to address the real-time stringent requirements of the lane change problem: drones will play the role of a central server that aggregates local parameters and road vehicles are clients that train local models. Particularly, each local model is based on the GL-DEAR model.

Chapter 5
Proposed DAFL Algorithm

5.1 Introduction

Based on the literature study in section 3.4, this chapter sheds the light on road active safety measurements using FL implemented in UAV assisted vehicular networks. Despite the great potential of deploying high-performance drones, drone battery life is a major concern, on one hand. On the other hand, road active safety is a critical real-time process that should be tackled in a tight time window in vehicular networks. To meet the mentioned issues, we adopt FL on the local vehicles, sending local updates to drone servers.

In order to select efficient clients, we introduce the concept of "reputation" for each client. To be more specific, we compute the reputation by client's local training reward, and the cosine similarity between its local update and the global update.

Moreover, a dynamic frequency adaptation framework is proposed to achieve the optimal trade-off between the road safety performance and drone energy consumption. The thresholds for the update frequency are calibrated according to road safety measurements (i.e., collision rate, risky and impolite driving time on the road) and drone energy consumption. Additionally, an accurate mathematical modeling based on M/G/1 multi-class preemptive queuing model is conducted in order to access the queuing time at the drone.

Compared to the existing work, our contributions are three-fold :

- We propose a drone-assisted federated deep reinforcement learning framework, DAFL, for vehicular network that takes into account road random risks and emergency vehicles.
- We propose a global model aggregation algorithm based on client's reputation. The latter is computed from the client's local training reward and the cosine similarity between the local update and the global update.
- We orient our efforts towards the communication delay analysis for three use cases (LCA, emergency vehicles, random road risk prevention) in vehicular networks. We perform a mathematical analysis using M/G/1 queue that evaluates vehicles delay and drone energy consumption.
- We elaborate an adaptive framework that tunes federated machine learning global update frequency according to road active safety performance parameters. This leads to an optimal trade-off between safety parameters and energy consumption.

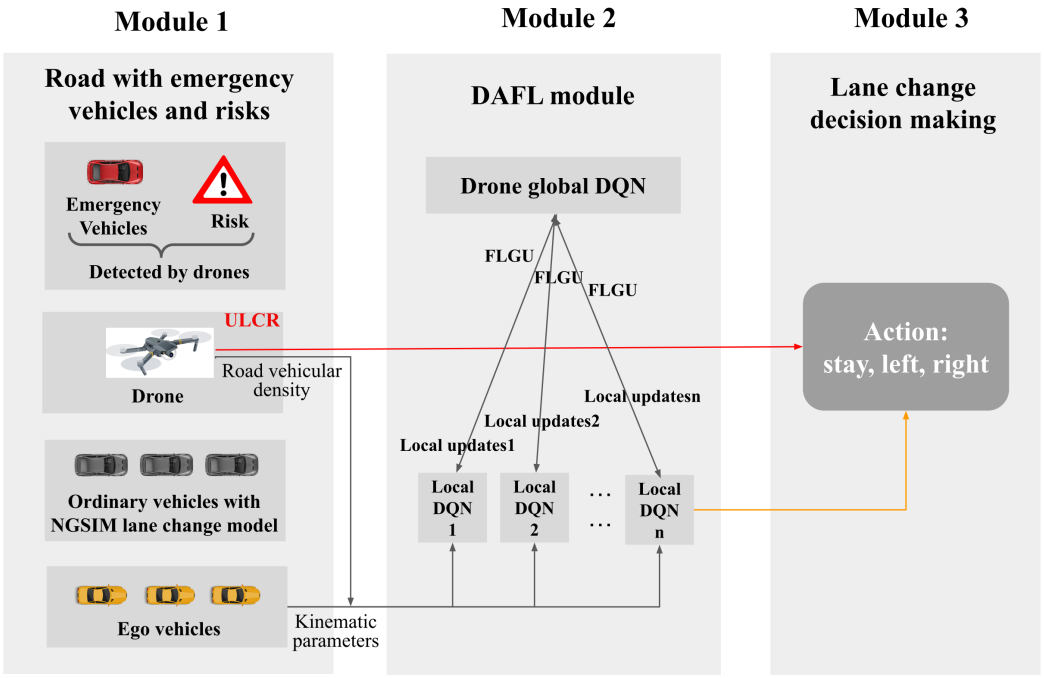

Figure 5.1: DAFL framework

The rest of the chapter is organized as followed: section 5.2 introduces the proposed algorithm, DAFL; section 5.3 presents the mathematical analysis of the E2E communication delay based on queuing theory; section 5.4 details the modeling of the drone energy consumption; section 5.5 shows the simulation results. Section 5.6 concludes the chapter with a discussion.

5.2 DAFL framework

Our proposed innovative DAFL framework trains a global lane change model that is aggregated from local vehicles in order to make safe and efficient lane change decisions. It is an enhanced version of GL-DEAR depicted in chapter 4. In fact, it integrates FL for the real time requirement and security issues.

The DAFL framework consists of 3 modules as illustrated in Fig. 5.1 and is explained in the following sections.

5.2.1 Module 1 Road with random risk and emergency vehicles

The first module of DAFL framework consists of three types of vehicles, i.e. ego vehicles, ordinary vehicles and emergency vehicles, where the ego vehicles are FL local clients and train local DRL models based on their own observations. In addition, drones play the role of central server in FL in addition to the global control by sending ULCRs, denoted by the red arrow from module 1 to module 3.

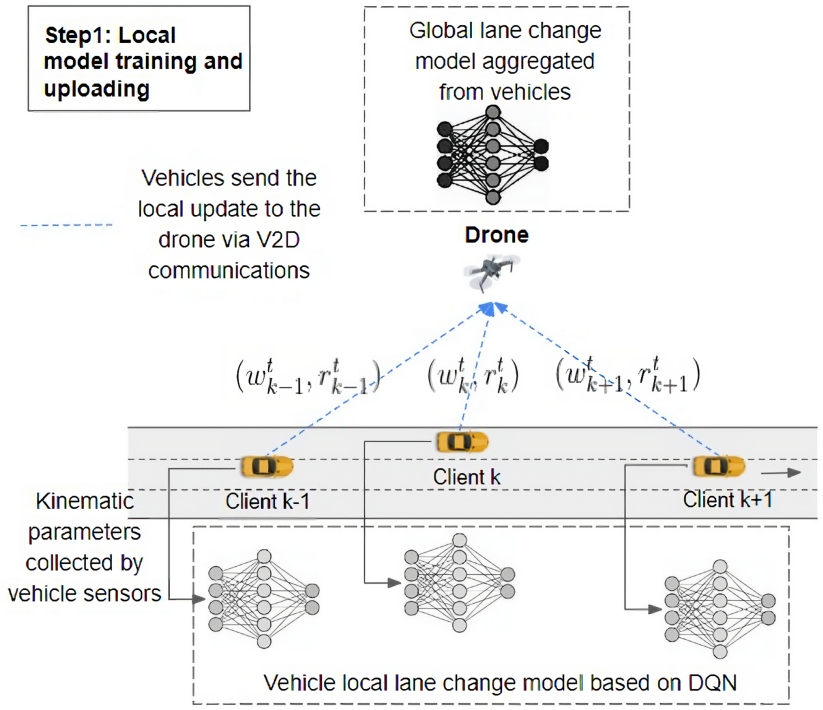

Figure 5.2: Step 1 of the DAFL algorithm: the local model training and uploading step

5.2.2 Module 2 DAFL module

Module 2 consists of the DAFL algorithm that manipulate the drone (central server) and selected vehicles (clients). Both the local model and the global model are DQNs with the same structure. The Federated Learning Global Update (FLGU) is the global model weights sent to clients at the end of each communication round. In the following sections, the DAFL algorithm, and the training for local DQNs will be detailed.

DAFL algorithm for drone server

As illustrated in Fig. 5.2 and 5.3, the DAFL consists of two main steps for each communication round: local model training and uploading, and global model aggregation and broadcasting.

At the beginning of every training simulation, the drone server first initializes the global model with w^0. Then, it selects a subset S_t from K clients to perform local training. Each selected client performs a number of local updates before sending the tuple (w_k^t, r_k^t) to the server; Where w_k^t and r_k^t denote the local update and the learning reward of the $k-th$ client at the $t-th$ communication round, respectively. Once the drone receives all the local updates, it will aggregate the local updates to a global update according to the following equation:

$$w_g^t = \sum_{k=1}^{n} \frac{p_k}{\sum_{j=1}^{n} p_j} w_k^t \tag{5.1}$$

where n is the number of local updates, p_k is the client reputation defined as in (5.3), which will be detailed afterwards. After the aggregation, the global update, w_g^t (i.e. FLGU), will be broadcasted to all the vehicle clients. The vehicle clients will update their model parameters

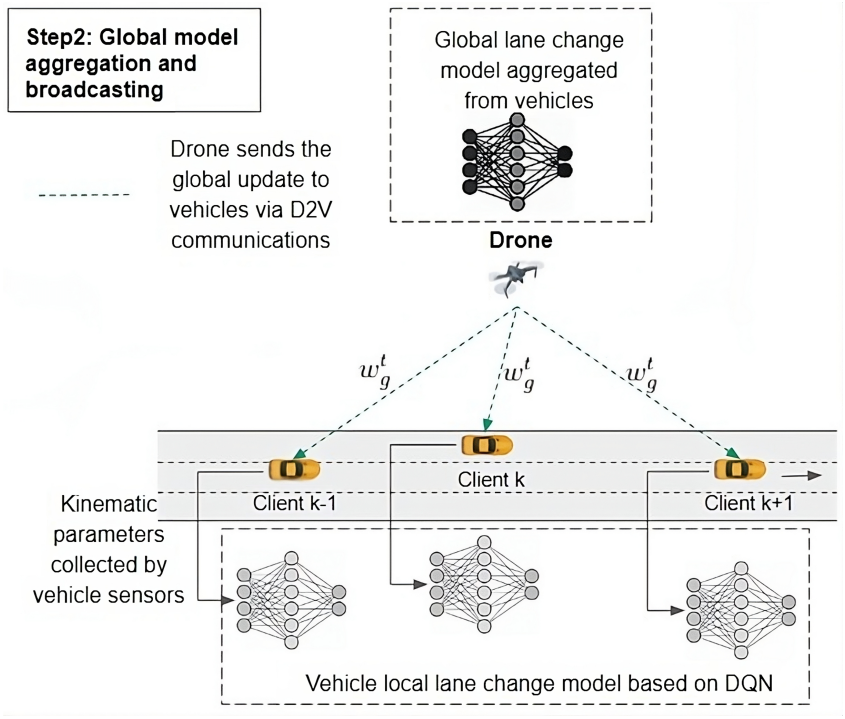

Figure 5.3: Step 2 of the DAFL algorithm: the global model aggregation and broadcasting step

with the new global model. Then, a new communication round begins with local vehicle clients performing local updates. The overall algorithm is formulated in Algorithm 3.

Deep reinforcement learning for vehicle clients

The considered scenario is presented in Fig. 5.4. On a 4-km circular highway, four drones hover over the highway with 1 km communication range [2]. Consequently, we divide the highway into four segments, each covered by one drone. Each drone hovers at different altitude to prevent collisions, and communicates only with vehicles located in its segment, as well as with other drones.

It is noteworthy that the drones send two types of messages:

- Urgent lane change request with high priority, denoted by ULCR.
- Federated learning global update with low priority, denoted by FLGU.

Our goal is to obtain a global lane change model that achieves the optimal trade-off between road active safety and drone's power consumption. It should be mentioned that both global model and local models are based on a DQN with the same structure. At the beginning of each communication round, the drone server selects a subset of vehicle clients to perform local update using their own observations. We adopt the same input feature as detailed in section 4.2.3 **State space**: the observation consists of 48 kinematic parameters consisting of the kinematic parameters of the vehicle client and its neighbors, the risk label of the current lane l_{risk}, (0 refers to no risk detection, 1 refers to risk detection), which is detected by vehicle sensors, in addition to the road vehicular density, which is calculated and provided by the drones.

Algorithm 3 DAFL algorithm

Input: Number of epoch per communication round, f.
Output: The trained global model
Drone server initializes the global model with w^0
Drone server selects a subset S_t of K clients to perform local updates
for each client $k \in S_t$ **in parallel do**
    for $t \leftarrow 0$ to T **do**
        if client k receives a new global model from the drone **then**
            Client k updates local model with the new global model
        end if
        if $t\%f \neq 0$ **then**
            Perform f local training epoch
        else
            Sends local update to the drone server
        end if
        if Drone server receives K local updates **then**
            Drone server aggregates the K local updates by (5.1)
            Drone server updates client reputation p_k according to (5.3)
            Drone server broadcasts the global update to the clients
        end if
    end for
end for

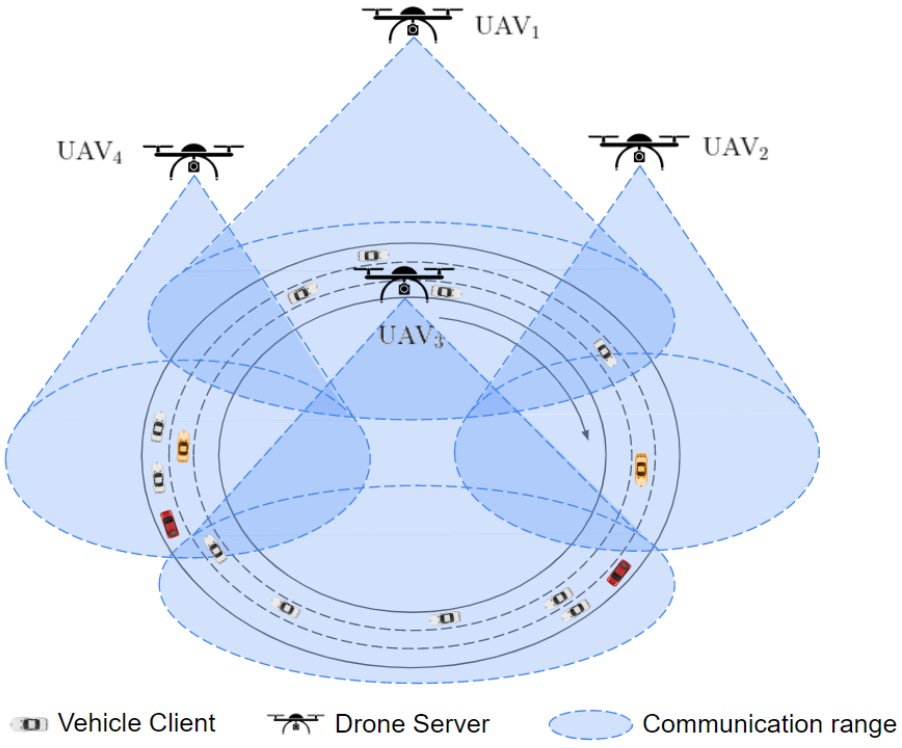

Figure 5.4: DAFL scenario

The action space is defined as the lane change behavior, denoted as $A_i^t = (-1, 0, 1)$, representing respectively lane changing to the left, staying in lane and lane changing to the right.

The reward function is the weighted sun of three parts: safety, efficiency and comfort, which are computed as in (4.2) and the following.

$$R_i^t = w_{safe} R_{safe} + w_{eff} R_{eff} + w_{comf} R_{comf}$$

where w_{safe}, w_{eff}, w_{comf} are the weights, R_{safe}, R_{eff} and R_{comf} are safety reward, efficiency reward and comfort reward during the DQN training. The safety reward considers four features: road vehicular density, collisions, road risk and emergency vehicles. The efficiency reward aims to increase speed and penalize frequent lane changes. And the comfort reward is designed to avoid hard brakes and hard accelerations. The computation of each reward is detailed in section 4.2.3 **Reward function** [70, 72].

At each step $t \in [0, T]$, each vehicle client $v_i, i \in [1, N_v]$ predicts the next movement with its local observation and local model, and performs the movement. Then, the tuple (observation, action, reward), i.e. $\left({O_i}^t, A_i^t, R_i^t\right)$ will be stored in its local memory buffer. After f local training epochs, the selected vehicle clients send their local model weights to the drone server. After receiving all the local weights, the server aggregates the local weights according to clients' reputations, p, defined as (5.3).

$$p_k = w_r \epsilon_k^r + w_c \epsilon_k^c \tag{5.2}$$

$$= w_r \frac{r_k}{\sum_{i=1}^{K} r_i} + w_c \frac{c_k}{\sum_{i=1}^{K} c_i} \tag{5.3}$$

where k is client index, ϵ_k^r is the local update efficiency of client k, computed by the reward of client k at current epoch, r_k. ϵ_k^c is the contribution to the global update, computed by the cosine similarity, c_k between the local update and the global update. The cosine similarity between two vectors, u and v are defined in (5.4). w_r and w_c are the weights of the two terms.

$$c(u, v) = \frac{\sum u_k v_k}{\sqrt{\sum_k u_k^2} \sqrt{\sum_k v_k^2}} \tag{5.4}$$

5.2.3 Module 3 Real-time lane change decision making

Module 3 is the real-time lane change decision making module based on local DQN. At each step, vehicle makes its lane change decision based on its local observation and local model. Noted that the local model is updated by the global model sent by drone server at the end of each communication round.

5.3 End-to-end delay modeling based on M/G/1 queue

This section provides an accurate mathematical modeling of the M/G/1 multi class preemptive queue. As previously mentioned, the drone tackles downlink messages that fall into two classes: ULCR, referred as class 1, and FLGU, referred as class 2.

Fig. 5.5 illustrates the queuing model for a single-drone assisted vehicular network. The D2V communications are represented by the grey dashed lines. Meanwhile, the queues of both of the two classes are shown at the drone side.

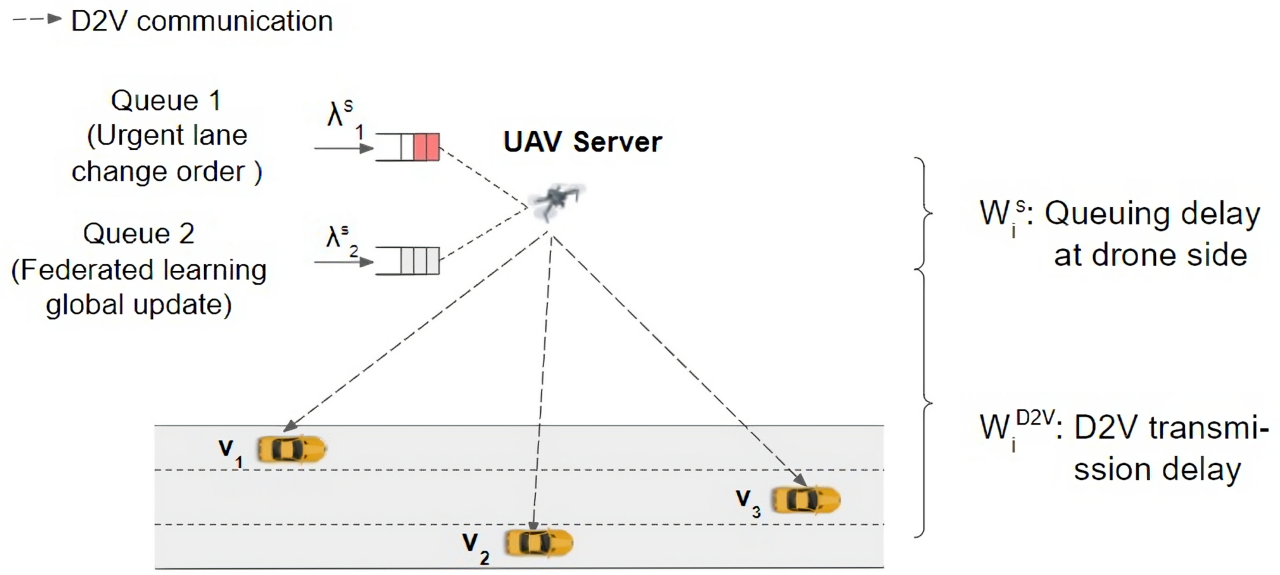

Figure 5.5: Illustration of the E2E delay analysis in DAVN.

The total E2E D2V delay of a class i message, denoted as $E[W_i]$, consists of two parts:

- Queuing delay at the drone side, W_i^s.
- D2V propagation delay, W_i^{V2D}.

Thus, we have

$$E[W_i] = E[W_i^s] + E[W_i^{D2V}] \leq T_i^* \tag{5.5}$$

where T_i^* is the optimal maximum waiting time for request class i, whose value is to work with. The computation delays at both the vehicle and the UAV side are neglected since these values are very small compared to the propagation delays. It should be noted that if a class i request is not processed by the UAV after T_i, it will be considered as expired.

In the following sections, we will derive the queuing delay and the propagation delay in detail.

5.3.1 Queuing delay at the drone side

As explained previously, we consider two types of message that can be transmit by the drone to vehicles: one is safety-related message with high priority, the other is vehicle state-related message with low priority. On the other hand, selected vehicle clients will send their local update weights to the drone after each f training epochs.

At the drone side, the service of a low priority request can be interrupted by the arrival of a high priority request. Thus, the queuing model is a priority queuing with preemption. In this case, the waiting time of the high priority requests, denoted as W_1, are not affected by the

low priority requests, and are only related to the arriving process and service process of class 1 request.

On the contrary, the waiting time of low priority requests, denoted as W_2, are affected by high priority requests, with an additional waiting time due to the arrival and interruption from a high priority request [81, 82].

We model the queuing delay of the safety message at the drones with a M/G/1 queue. The arriving process follows a Poisson process while the service time follows an exponential distribution. What is more, the service times for different safety messages are independent and identically distributed.

We define the following variables for the drone:

- ρ_i: the occupation rate of a class i message,
- λ_i: the arrival rate of a class i request,
- $E[B_i]$: the mean service time at the drone of a class i request,
- $E[W_i]$: the mean waiting time in the queue of a class i request,
- $E[R_i]$: the mean residual service time of a class i message,
- $E[S_i]$: the mean sojourn time of a class i request ($E[S_i] = E[B_i] + E[W_i]$),
- $E[L_i]$: the average number of requests of class i waiting in the queue.

Thus, for the total incoming traffic at the drone side, we have the following:

$$\lambda = \sum_{i=1}^{n} \lambda_i \tag{5.6}$$

$$E[B] = \sum_{i=1}^{n} \frac{\lambda_i}{\lambda} \cdot E[B_i] \tag{5.7}$$

$$\rho = \lambda \cdot E[B] \tag{5.8}$$

Mean waiting time of high priority requests

The average of W_1 can be expressed as followed:

$$E[W_1] = E[L_1]E[B_1] + \rho_1 E[R_1] \tag{5.9}$$

where L_1 denote the number of high priority request waiting in the queue. According to Little's law we have

$$E[L_1] = \lambda_1 E[W_1] \tag{5.10}$$

Combining the two equations yields

$$E[W_1] = \frac{\rho_1 E[R_1]}{1 - \rho_1} \tag{5.11}$$

Since we have

$$E[R_1] = \frac{E[B_1^2]}{2E[B_1]} \tag{5.12}$$

Equation (5.11) becomes

$$E[W_1] = \frac{\rho_1}{2(1-\rho_1)} \cdot \frac{E[B_1^2]}{E[B_1]} \tag{5.13}$$

The sojourn time is then

$$E[S_1] = E[W_1] + E[B_1] = \frac{\rho_1}{2(1-\rho_1)} \cdot \frac{E[B_1^2]}{E[B_1]} + E[B_1] \tag{5.14}$$

Mean waiting time of low priority requests

As explained before, the waiting time of low priority requests can be expressed as

$$E[W_2] = E[B_2] + E[W_+]$$

The request has to wait for the sum of the service times of all previous requests with the same or higher priority as well as the remaining service time of the request in service. Consequently,

$$E[B_2] = \sum_{j=1}^{2} E[L_j]E[B_j] + \sum_{j=1}^{2} \rho_j E[R_j] \tag{5.15}$$

On the other hand, the W_+ is related to all the higher priority requests arriving during its waiting time and service time. This leads to

$$E[W_+] = \lambda_1 E[W_1]E[B_1] \tag{5.16}$$

Applying Little's law

$$E[L_2] = \lambda_2 E[W_2]$$

we have

$$E[W_2] = \frac{\sum_{j=1}^{2} \rho_j E[R_j]}{(1-(\rho_1+\rho_2))\,(1-\rho_1)} \tag{5.17}$$

The mean sojourn time $E[S_2]$ of a class i customer follows from $E[S_2] = E[W_2] + E[B_2]$, yielding

$$E[S_2] = \frac{\sum_{j=1}^{2} \rho_j E[R_j]}{(1-(\rho_1+\rho_2))\,(1-\rho_1)} + E[B_2] \tag{5.18}$$

Since we have

$$E[R_i] = \frac{E[B_i^2]}{2E[B_i]}, \tag{5.19}$$

Equation (5.20) finally becomes

$$E[S_2] = \frac{1}{(1-(\rho_1+\rho_2))\,(1-\rho_1)} \cdot \sum_{j=1}^{2} \rho_j \frac{E[B_i^2]}{2E[B_i]} + E[B_2] \tag{5.20}$$

5.3.2 D2V propagation delay

The D2V propagation delay of a class i request is calculated as follows:

$$E[W_i^{D2V}] = \frac{d_i}{r_i} \tag{5.21}$$

where d_i is the euclidean distance between the UAV and the vehicle, and r_i is the achievable propagation rate.

5.4 Drone Energy Consumption Model

The energy consumed by the UAV i in order to perform a complete data transfer is composed of three components as shown in equation (6.4):

- the communication energy that is used for the data transfer from the UAV to vehicles clients and other UAVs, E_i^t,
- the computing energy for global model aggregation, E_i^c,
- the propulsion energy of the UAV to hover over the highway, E_i^m.

$$\begin{aligned} E_i &= E_i^t + E_i^c + E_i^m \\ &= \sum_{j \in \mathbb{V}} E_{i,j}^{D2V} + E_i^c + E^m \end{aligned} \tag{5.22}$$

where $\mathbb{V}$ is the set of all vehicle clients in the communication range of the UAV i that participate in the federated learning, $\mathbb{U}$ denotes the set of all UAVs. One can see that we only consider the downlink D2V communication. In the following sections, each of the elements will be analysed.

5.4.1 Communication energy

In order to model the communication energy consumption of our scenario, we first look at a simple scenario where the UAV i is communicating with vehicle j, as illustrated in Fig. 5.6. h_i is the height of the UAV i, R is the communication range of the UAVs. $d_{i,j}^{euc}$ and $d_{i,j}^{hor}$ denote the Euclidean distance and the horizontal distance between the UAV i and vehicle j, respectively. The D2V path loss is also known as air-to-ground path loss. In [83], the probability of having a line-of-sight link between the UAV i and vehicle j is formulated as followed:

$$P_{i,j}^{D2V}(LoS) = \frac{1}{1 + a \, \exp\left(-b\left(\theta_{i,j} - a\right)\right)} \tag{5.23}$$

where a and b are environmental constant depending on rural or urban areas; θ is the elevation angle between UAV i and vehicle j, and it is equal to $\arctan(\frac{h_i}{d_{i,j}^{hor}})$; h_i is the height of UAV i from ground level; $d_{i,j}^{hor}$ is the horizontal distant between the UAV i and the vehicle j.
Thus, the probability of non-line-of-sight loss is calculated as:

$$P_{i,j}^{D2V}(NLoS) = 1 - P_{i,j}^{D2V}(LoS) \tag{5.24}$$

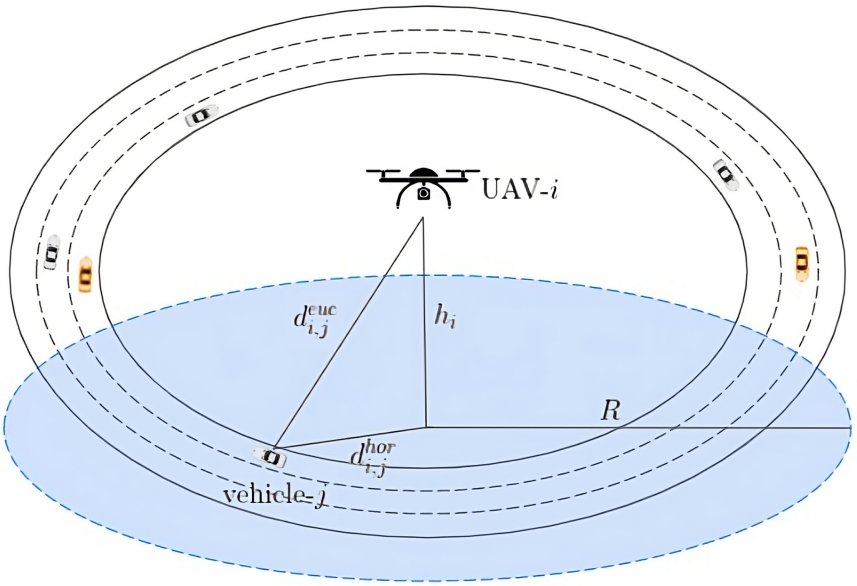

Figure 5.6: Scenario where the UAV i is communicating with vehicle j

According to [84], the path losses with LoS and No- LoS links between UAV i and vehicle j can be formulated as:

$$PL_{i,j}^{D2V}(LoS) = 20\ \log_{10}\left(\frac{4\pi f_c d_{i,j}^{euc}}{c}\right) + \eta_{LoS} \tag{5.25}$$

$$PL_{i,j}^{D2V}(NLoS) = 20\ \log_{10}\left(\frac{4\pi f_c d_{i,j}^{euc}}{c}\right) + \eta_{NLoS} \tag{5.26}$$

where η_{LoS} and η_{NLoS} are the mean additional losses for LoS and No Line-of-Sight (NLoS) links, c is the speed of light, and $d_{i,j}^{euc} = \sqrt{h_i^2 + \left(d_{i,j}^{hor}\right)^2}$ is the euclidean distance between the UAV i and vehicle j.

As a result, the average path loss of the D2V communication between UAV i and vehicle j can be computed as followed:

$$\begin{aligned}
&PL^{D2V}(i,j) \\
&= P_{i,j}^{D2V}(LoS) PL_{i,j}^{D2V}(LoS) \\
&+ P_{i,j}^{D2V}(NLoS) PL_{i,j}^{D2V}(NLoS) \\
&= 20\ \frac{1}{1 + a\ \exp\left(-b\left(\theta_{i,j} - a\right)\right)} \left[\log_{10}\left(\frac{4\pi f_c d_{i,j}^{euc}}{c}\right) + \eta_{LoS}\right] \\
&+ 20\ \ \left(1 - P_{i,j}(LoS)\right) \left[\log_{10}\left(\frac{4\pi f_c d_{i,j}^{euc}}{c}\right) + \eta_{NLoS}\right] \\
&= \frac{\eta_{LoS} - \eta_{NLoS}}{1 + a\ \exp\left(-b\left(\theta_{i,j} - a\right)\right)} \\
&+ 20\ \log_{10}\left(d_{i,j}^{euc} \sec(\theta_{i,j})\right) + 20\ \log_{10}\left(\frac{4\pi f_c}{c}\right) + \eta_{NLoS}
\end{aligned}$$

And the channel gain is given by

$$G^{D2V}(i,j) = \frac{1}{PL^{D2V}(i,j)} \tag{5.27}$$

Consequently, the Signal-to-Noise Ratio, $\text{SNR}_{i,j}$ and the achievable data rate $C_{i,j}$ in bits per second (bps) of the D2V communication are presented in Equation. (5.28) and (5.29) [85–87]:

$$\text{SNR}_{i,j}^{D2V} = \frac{p_i^{trans} G_{i,j}^{D2V}}{\sum_{n \in \mathbb{N}_{\text{int}}} p_n^{trans} G_{n,j}^{D2V} + N_0} \tag{5.28}$$

where p_i^{trans} is the transmit power of UAV i, $\mathbb{N}_{\text{int}}$ is the set of possible interfering UAVs, N_0 is the noise power.
According to Shannon's theorem, the achievable rate of the D2V communication is:

$$C_{i,j}^{D2V} = B * \log_2\left(1 + \text{SNR}_{i,j}^{D2V}\right) \tag{5.29}$$

where B is the bandwidth for the D2V communication. The energy consumption of the UAV transmitting the message is [88]:

$$E_{i,j}^{D2V} = \frac{S_i}{C_{i,j}^{D2V}} \, p_i^{trans} \tag{5.30}$$

where S_i is the size of the message that UAV i is going to send.

5.4.2 Computation energy

The energy for computation refers to the energy for the global model aggregation, denoted as E^c. It is calculated as (5.31) in [54].

$$E^c = kC\phi^2 \sum_{i=1}^{I} S(w_i) \tag{5.31}$$

where k is the energy consumption coefficient related to the computing system; C is the computing cycles needed for each data bit; ϕ is the frequency of drone's CPU clock; and $S(w_i)$ is the packet size of weight w_i sent to client i.

5.4.3 Mobility energy

According to [89, 90], the motion power model P_i of UAV i of speed V_i is represented in the following equations.

$$P_i = P_0\left(1 + \frac{3V_i^2}{U_{\text{tip}}^2}\right) + P_1\left(\sqrt{1 + \frac{V_i^4}{4v_0^4}} - \frac{V_i^2}{2v_0^2}\right)^{1/2} \tag{5.32}$$

$$+ \frac{1}{2} d_0 \rho s A V_i^3 \tag{5.33}$$

$$P_0 = \frac{\delta}{8} \rho s A \Omega^3 R^3 \tag{5.34}$$

$$P_1 = (1+k)\frac{W^{3/2}}{\sqrt{2\rho A}} \tag{5.35}$$

In fact, P_0 and P_1 are two power constants representing the blade profile and the induced power levels in hovering status, respectively. U_{tip} denotes the tip speed of the rotor blade, v_0 is known as the mean rotor induced velocity in hovering, d_0 and s are the fuselage drag ratio and rotor solidity, respectively, ρ and A denote the air density and rotor disc area, respectively, δ is the profile drag coefficient, and V is the forward speed [89, 90]. The parameter values are defined in Table. 1 in [89]. Thus, the energy consumption to move from location $M(x_1, y_1, z_1)$ to location $M'(x_2, y_2, z_2)$ can be calculated by Equation. (5.37):

$$E_i(M, M') = \frac{d_{MM'}}{V_i} P_i \tag{5.36}$$

$$= \frac{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2}}{V_i} P_i \tag{5.37}$$

In the case of stationary hovering, V_i is zero and the drone remains stationary. Hence, $P_i = P_0 + P_1$. The hovering energy is

$$E_{hov} = T_{hov} P_{hov} = T_{hov}(P_0 + P_1), \tag{5.38}$$

where T_{hov} is the hovering time.

5.5 Performance Evaluation of the DAFL Framework

5.5.1 Simulation scenario

Extensive simulation batches are conducted with SUMO [25]. The simulation parameters are depicted in Table 5.1.

Table 5.1: Simulation parameters

Parameter	Value	Reference	Parameter	Value	Reference
λ_1	2.5	[91]	a	14.39	[85]
λ_2	Vehicle client number	[91]	b	0.13	[85]
$E[B_1]$	4.763×10^{-7}	[92]	f_c	2.4GHz	[2]
$E[B_1^2]$	2.269×10^{-13}	[92]	η_{LoS}	1dB	[85]
$E[B_2]$	2.977×10^{-8}	[92]	η_{NLoS}	20dB	[85]
$E[B_2^2]$	0	[92]	p_i^{trans}	280mW	[2]
B	100MHz	[2]	N_0	-174dB/Hz	[87]

We trained global models on a 4-km circular highway that consists of three lanes. It's noteworthy that the considered highway includes random road risks: construction works and vehicle collisions. Furthermore, we assume that 20% of the vehicles are emergency vehicles (ambulances and police cars). In addition, 3% of the vehicles are set with aggressive behaviors such as exceeding the speed limit and occupying the left-most lane for a long time. The same as our previous work, the training and testing are performed with Gym module in Python interfaced

with TraCI module in SUMO [93]. In the simulation, the vehicles' states are updated every 0.4 second, the lane change time is set to 2 seconds. However, while performing lane changing, the new lane change decisions are ignored to prevent conflict.

It should be pointed out that, in our designed platform, we consider that there are two states for the vehicles: completely autonomous and partial autonomous. Vehicle switches between these two states according to environment and traffic density. In urban scenarios, the vehicle adopts the human-driven mode where humans can interfere with the lane change decision. In highway scenarios, the vehicle adopts the autonomous mode where the machine learning model makes the lane change decision. In case of conflicts between human and machine learning lane change recommendation, the human decision is adopted since he/she is responsible for the road active safety.

In order to investigate the impact of global model update frequency, we train the global models with different model update frequencies, from 10 to 100 epochs per communication round with 500 epochs and 64 steps per epoch. Local and the global models are DQNs with the same structure consisting of 64 (resp. 128, resp. 64) nodes on the first (resp. second, resp. third) hidden layer.

At the drone level, the computation of the global weights relies on the weighted weights method as described in Equation. (5.1). The conducted simulations were performed with four, six and eight local clients, denoted by ww4, ww6 and ww8. For each of the trained models, we considered three different traffic densities: sparse traffic with 50 vehicles, medium traffic with 150 vehicles and dense traffic with 250 vehicles on the road.

The average training reward using 4 agents are shown in Fig. 5.7. In more detail, the rewards are the moving average of 500 epochs of each agent [94]. One can see that the training passes through an exploitation phase before converging to a stable training phase. The same behavior is observed for six agents and eight agents. It should be noticed that the maximum average reward is reached at around 1700 training epochs with 32000 simulation steps.

The performance parameters are road collision rate, the number of lane changes, vehicle average speed, mean risky time and mean blocking time, which are the same as described in chapter 4, in addition to the average total delay of vehicle clients and drone energy consumption. Recall that risky time and blocking time in second are the total time that the vehicle drive near a road risk and in front of an emergency vehicle without giving way.

It should be noted that each model is trained and tested with 32000 epochs with a laptop with Intel(R) Core(TM) i7-1185G7. The training, executed by drones and based on federated deep reinforcement learning, aims at storing local policies on vehicular agents. The drones perform the training once when needed.

5.5.2 Performance analysis

The performance of the three models are depicted in figures 5.8 to 5.28.

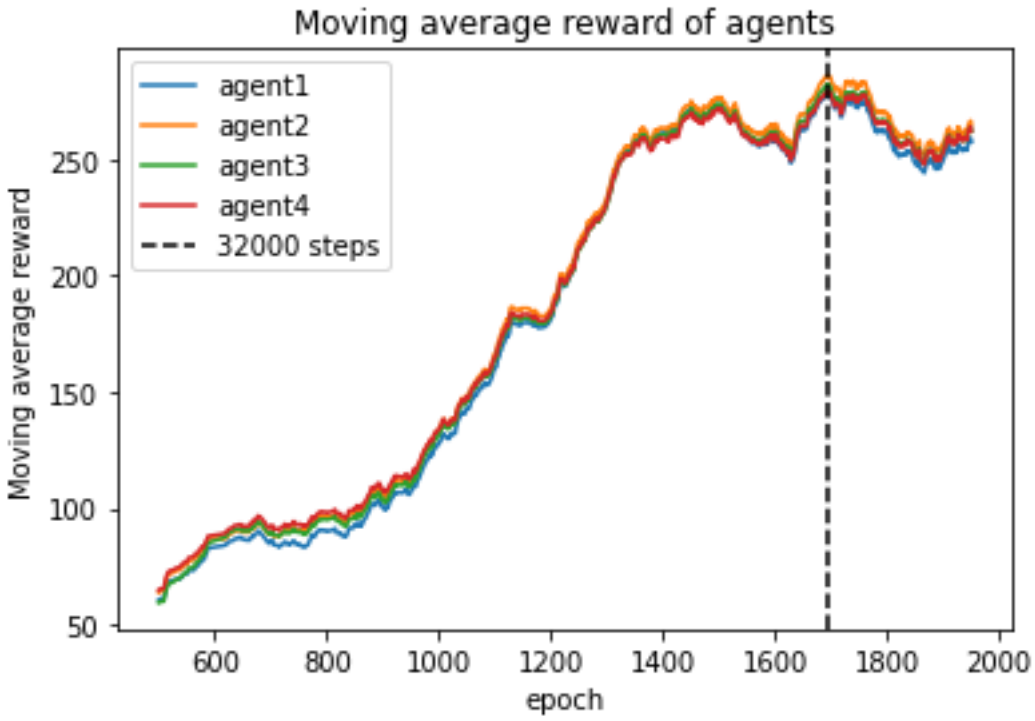

Figure 5.7: Moving average reward of agents

In the figures, we denote by nv the road vehicular density, and by epoch per communication round the local training epoch number before uploading the weights and rewards to the server.

Fig. 5.8 to 5.11 illustrate the lane change number, average speed, risky time and blocking time obtained with the ww4 model. One can see that with the same number of epoch per communication round, the average lane change number, risky time and blocking time increase when the traffic density increases. On the contrary, the average speed is decreased. This is expected since the more vehicles on the road, the more lane changes the vehicle should perform to continue the trip, the lower the average speed is and the longer the vehicle drives on risky lane. Furthermore, in dense traffic, there will be more emergency vehicles drive on the road, thus leads to longer blocking time.

On the other hand, with the increase of the number of epoch per communication round, it can be found that the lane change number, average speed, risky time and blocking time increase. Indeed, the higher the epoch number is, the fewer the global aggregation number is. This will incur fewer global updates that will lead to outdated global weights, which degrades the model performance (i.e. more inefficient lane changes, higher risky time and blocking time). Further, as a result of the lane changes, the vehicle is able to get higher speed during the trip.

Fig. 5.12 to 5.19 exhibit the performance parameters obtained with the ww6 model and ww8 model. The reader can notice that the risky time and blocking time increase as the number of epoch per communication round increases. However, the lane change number decreases when the number of epoch per communication round is 100 compared to 75 epochs. One possible reason is that with few global updates, which aggregates information from clients, the model is not able to learn the environment clearly. In other word, the global model is not well trained

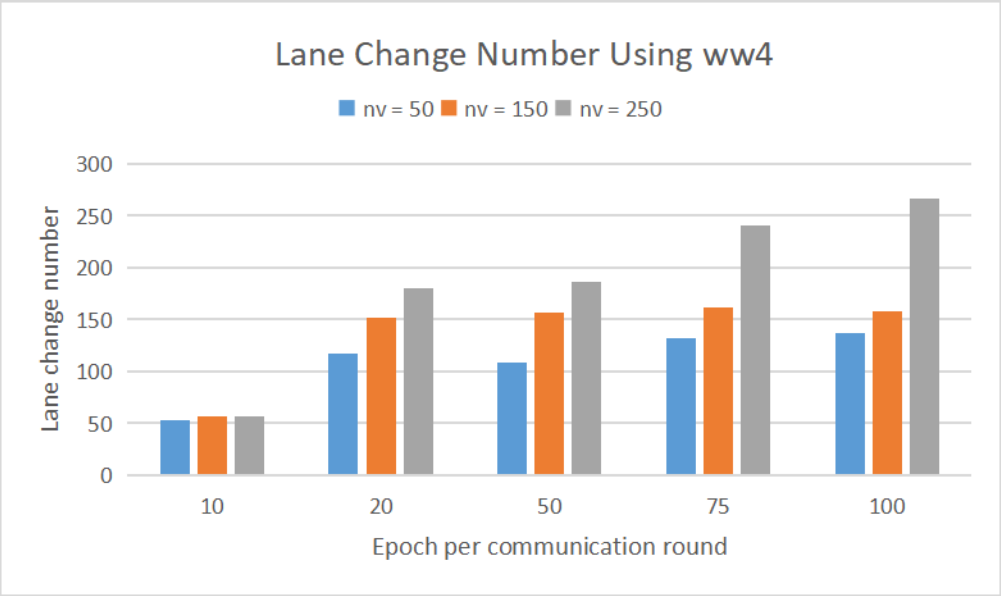

Figure 5.8: Lane change number of global model with 4 clients (ww4)

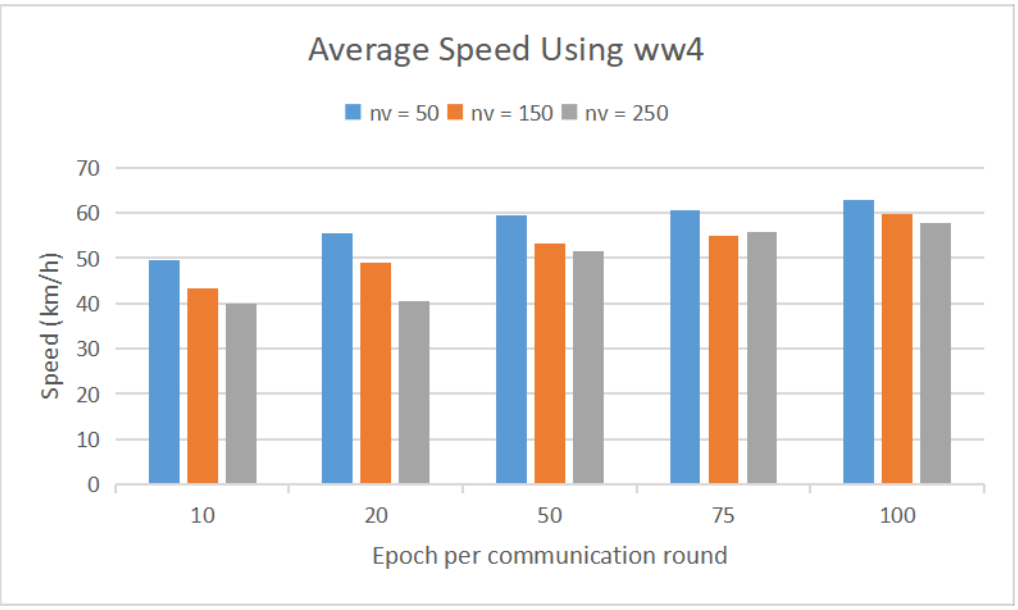

Figure 5.9: Average speed of global model with 4 clients (ww4)

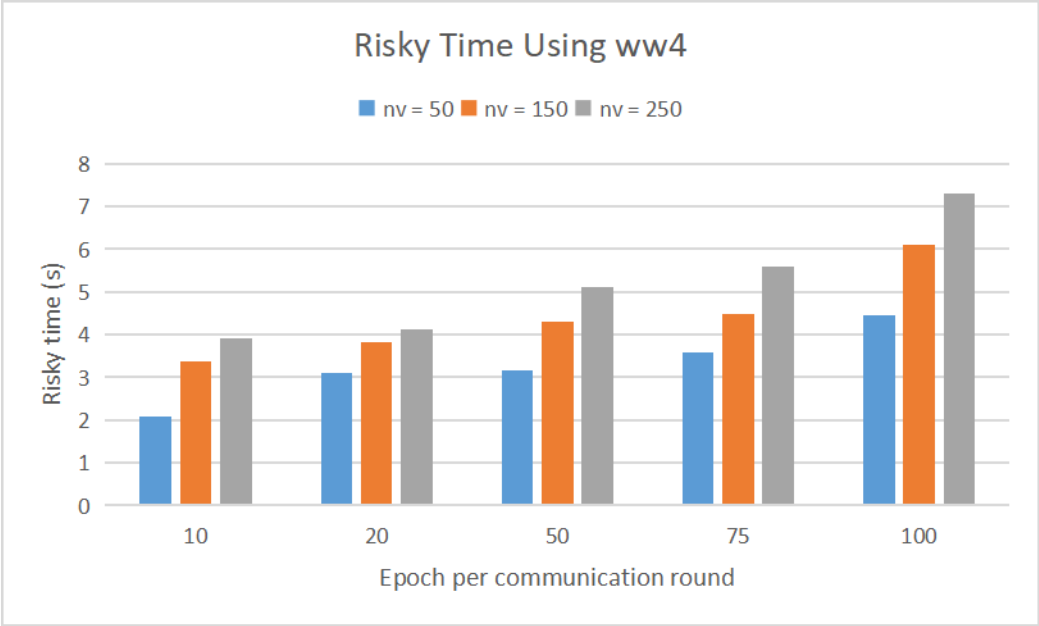

Figure 5.10: Risky time of global model with 4 clients (ww4)

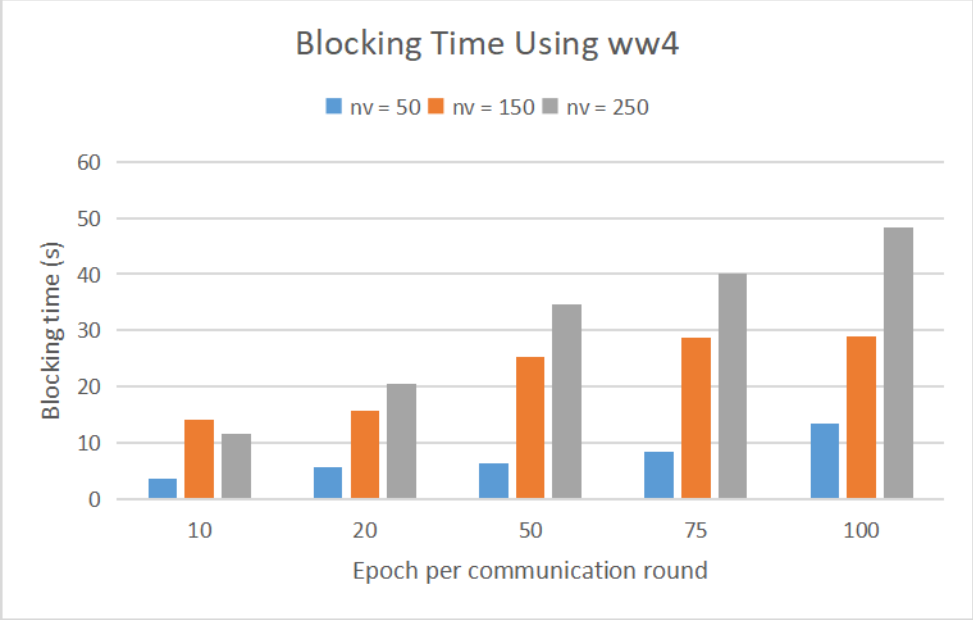

Figure 5.11: Blocking time of global model with 4 clients (ww4)

when local updates number is small. Consequently, it tends to be less active, leading to few lane changes. This further harms the model performance, which can be proven from the longer risky time and blocking time.

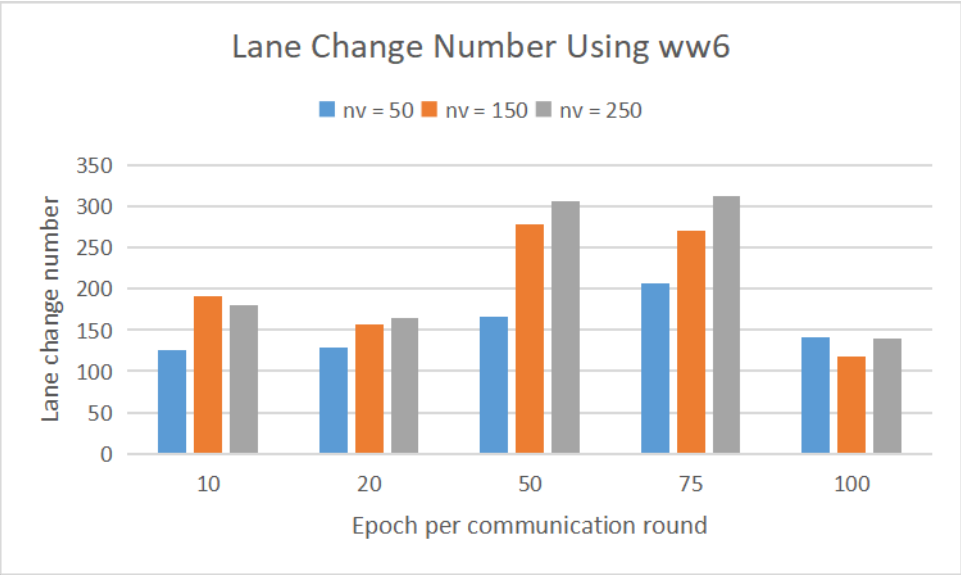

Figure 5.12: Lane change number of global model with 6 clients (ww6)

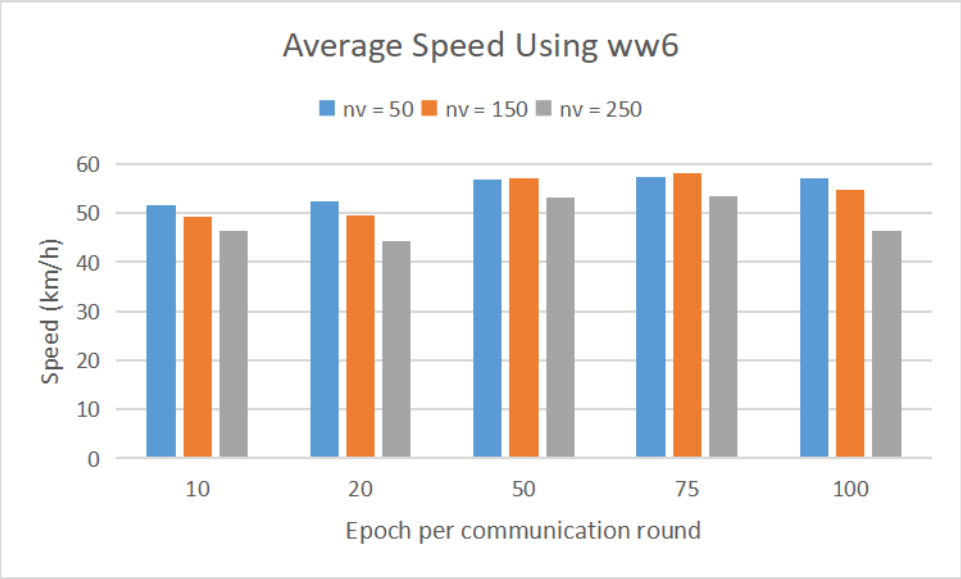

Figure 5.13: Average speed of global model with 6 clients (ww6)

On the other hand, we compare the performance of the three model, as shown in Fig. 5.20 to 5.23. Generally, we can find that the lane change number, average speed, risky time and blocking time increase as the number of epoch per communication round increases. As a matter of fact, when the number of epoch per communication round is small, there will be more global updates, which leads to a more efficient global model. Contrarily, as previously analysed, when the number of epoch per communication round is high, there will be few global updates, leading to an outdated global model and performance fluctuations.

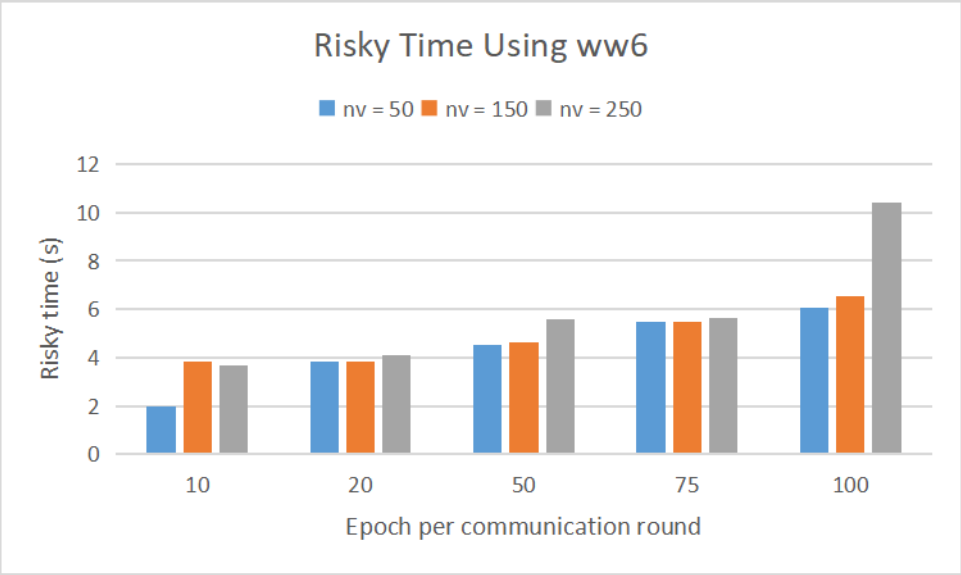

Figure 5.14: Risky time of global model with 6 clients (ww6)

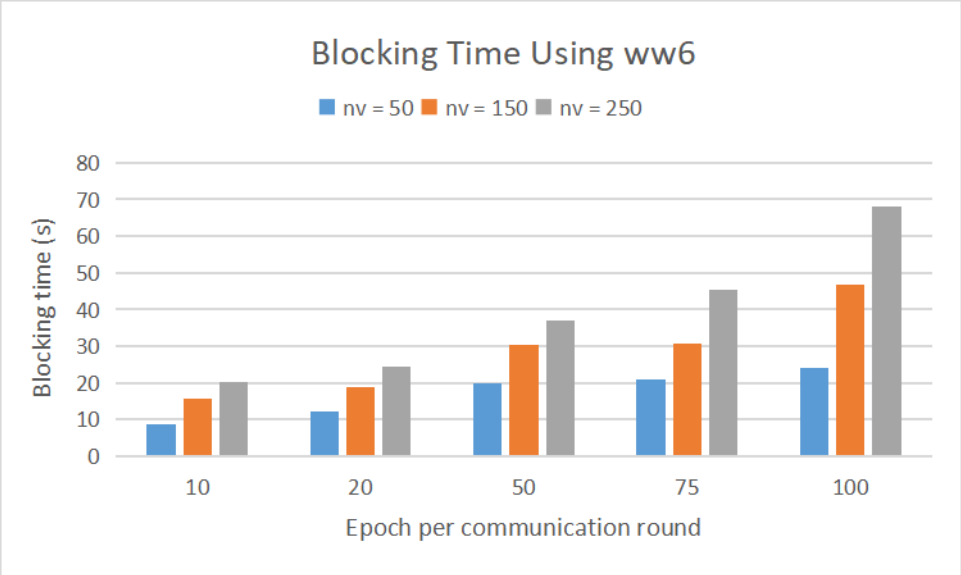

Figure 5.15: Blocking time of global model with 6 clients (ww6)

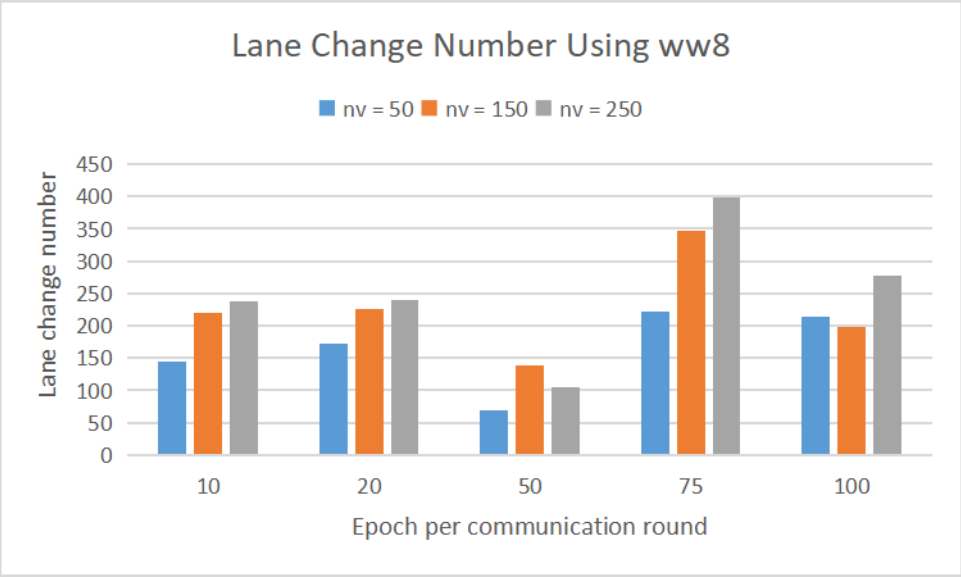

Figure 5.16: Lane change number of global model with 8 clients (ww8)

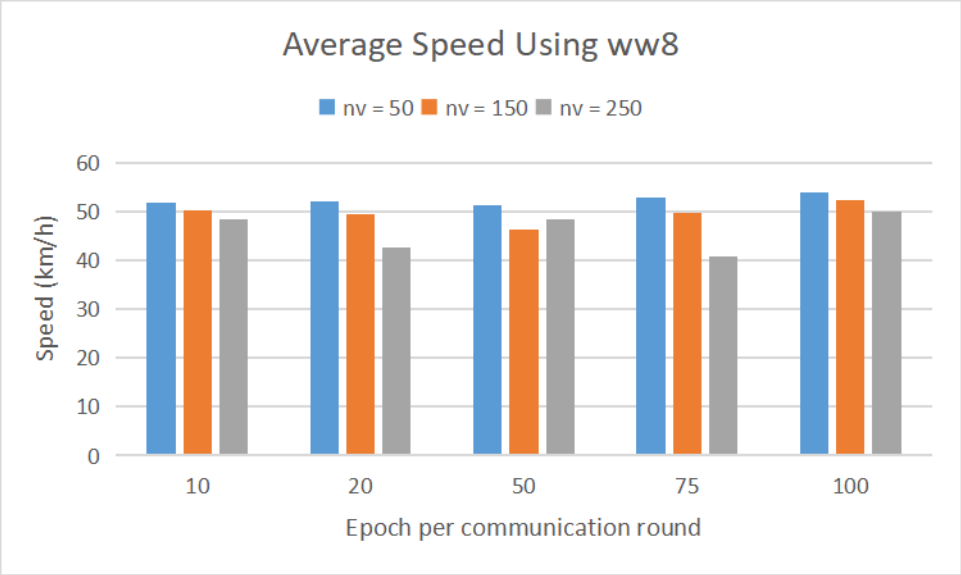

Figure 5.17: Average speed of global model with 8 clients (ww8)

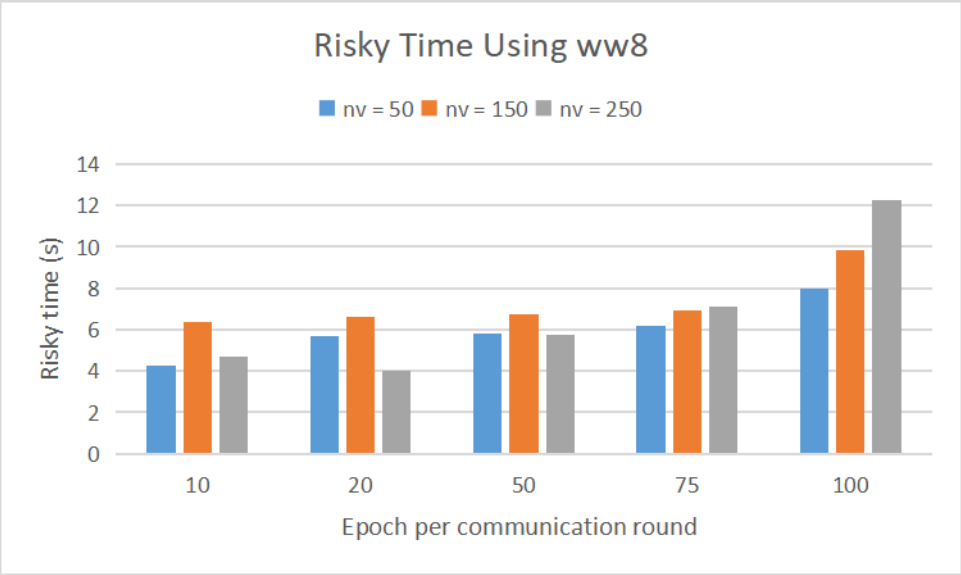

Figure 5.18: Risky time of global model with 8 clients (ww8)

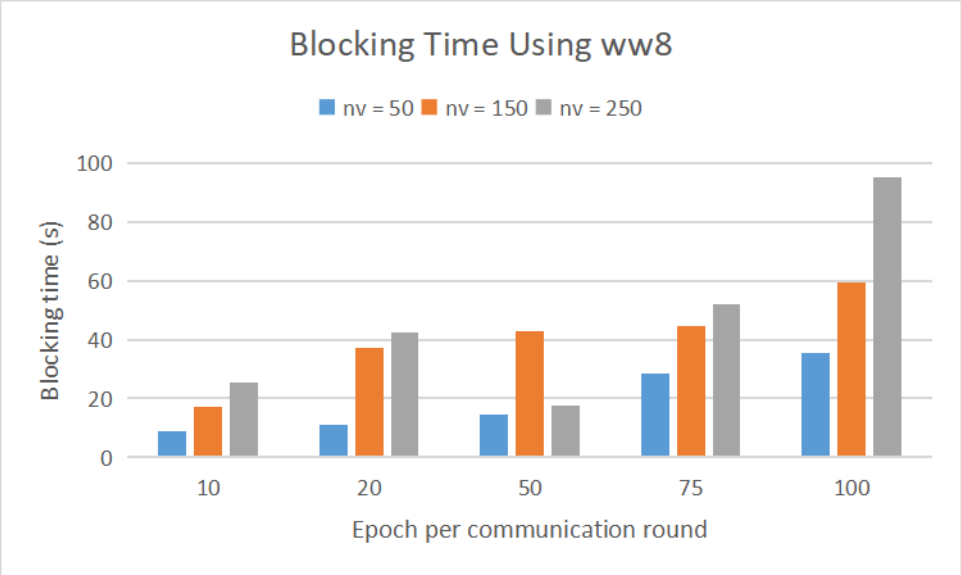

Figure 5.19: Blocking time of global model with 8 clients (ww8)

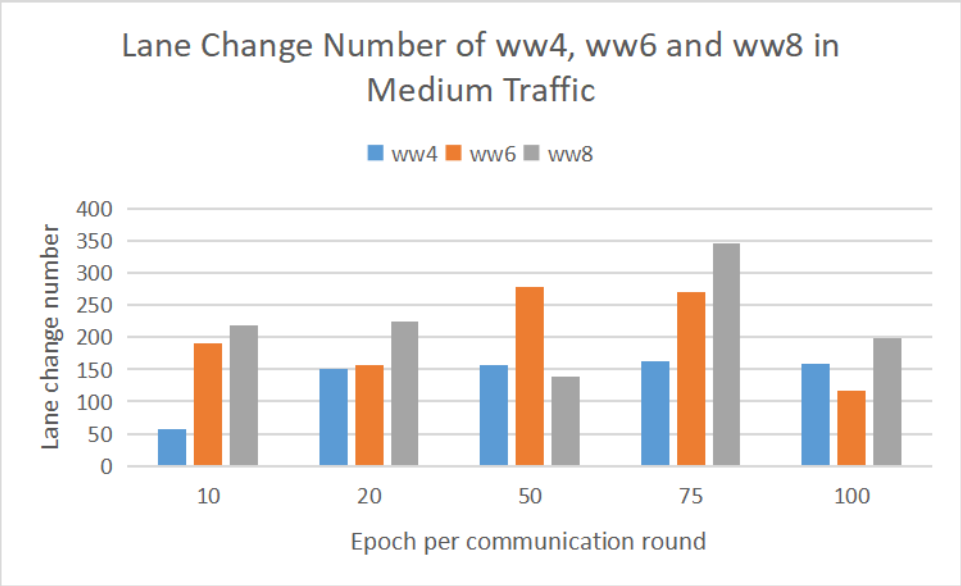

Figure 5.20: Comparison of lane change number of ww4, ww6 and ww8 in medium traffic

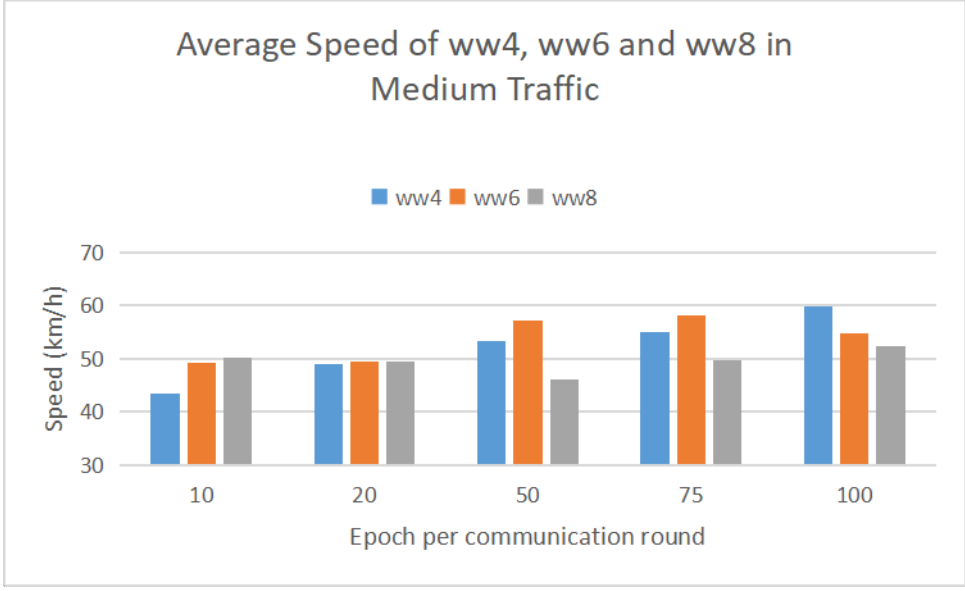

Figure 5.21: Comparison of average speed of ww4, ww6 and ww8 in medium traffic

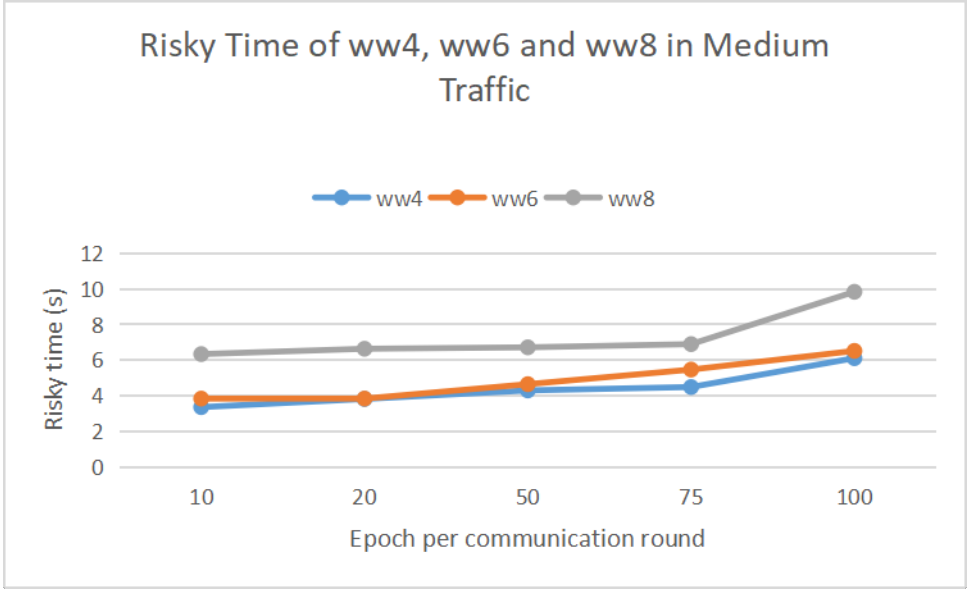

Figure 5.22: Comparison of risky time of ww4, ww6 and ww8 in medium traffic

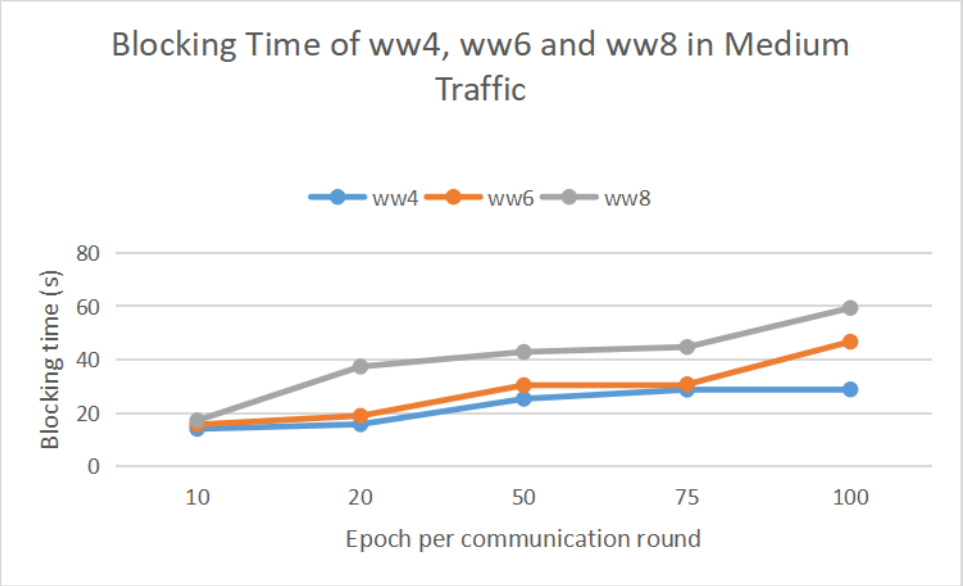

Figure 5.23: Comparison of blocking time of ww4, ww6 and ww8 in medium traffic

Fig. 5.24 depicts the average E2E delay and the total drone power consumption. It is noteworthy that the delay consists of the queuing delay at the drone and the propagation delay from the drone to the local client. One can see that the average total delay of each client is bounded by 0.7 second, which is acceptable for the critical real-time lane change use case. In addition, the total drone power consumption is lower than 0.62 kWh, which is a quite satisfying result. In fact, to support this energy consumption constraint for the whole mission, the required battery capacity BC for a LiPo battery of 40V Voltage is computed as followed:

$$BC = \frac{0.64kWh}{40V \times 32000 \times 0.4second} = \frac{640Wh}{40V \times 3.56h} \tag{5.39}$$

$$\approx 4.49Ah = 4490mAh. \tag{5.40}$$

The existing commercial drone batteries, such as DJI's Intelligent Flight Battery TB65 [95], meet the requirements for this battery. Thus, for the drone training mission duration, there can be enough electricity drawn from battery storage to provide 0.62 kWh power consumption. This value shows the potential for achieving sustained mission duration within certain energy constraints. Moreover, as an enhancement of our protocol, one can define an energy consumption threshold for the whole drone mission. When energy consumption surpasses the predefined threshold, it is imperative for the drone to adapt the flight speed and communication frequency. Such adaptive mechanisms serve to harmonize operational requirements with energy efficiency, thereby enhancing the overall performance and longevity of the system.

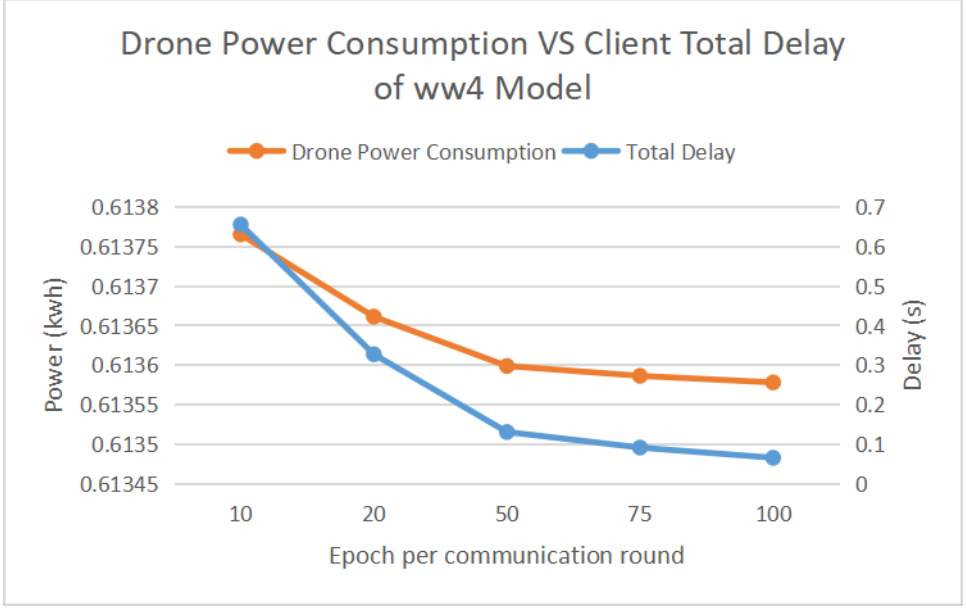

Figure 5.24: Drone power consumption and vehicle's delay

Specifically, the power for mobility, computation and communication are further compared in Fig. 5.25. One can see that the computation power and communication power are much less than the mobility power. Actually, drone's mobility is the most power-consuming part in the total power consumption of a drone. According to the numerical results, 99.97% of the energy is used for mobility.

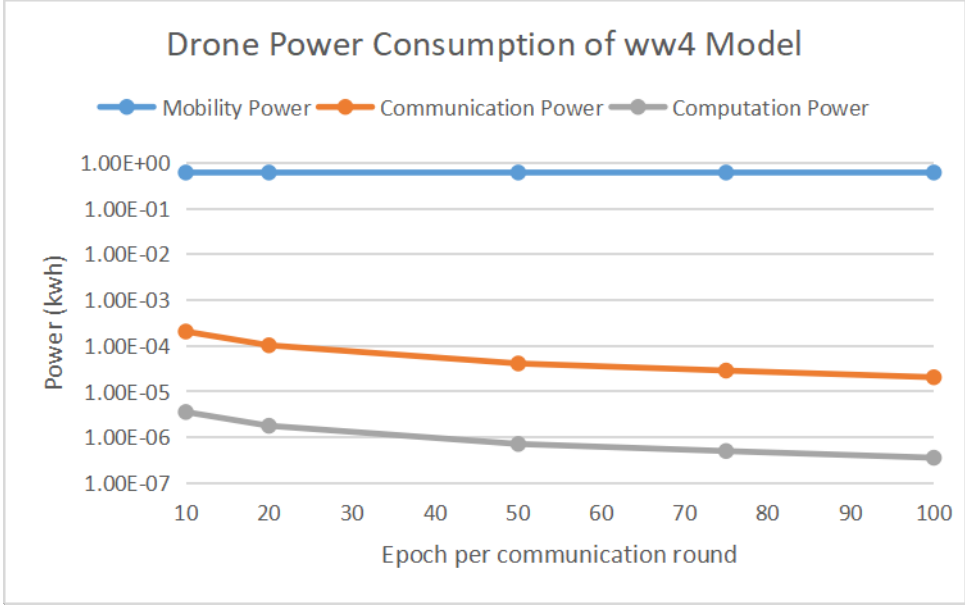

Figure 5.25: Drone's power consumption, including mobility power, communication power and computation power

In figures 5.26 to 5.28, we sheds the light on the trade-off between safety parameters (risky time and blocking time) and total power consumption. From the three figures, one can obtain the best trade-off safety and power consumption by the intersection point of risky time and power consumption, as well as the intersection point of blocking time and power consumption. For all of the three models, the best trade-off is achieved with 30 epochs per communication round for risky time. With this understanding, the drone server can dynamically adapt local training epoch number according to the road safety and its battery life.

Inspired by the DAFL performance analysis, we can draw the following conclusions:

- The best safety parameters were retrieved with four local clients as compared to six clients and eight clients.
- Road active safety stakeholders should tune the epoch per communication round according to the balance between risky time (or blocking time) and the total power consumption.
- The federated machine learning succeeds to respect the stringent delay requirements of the critical real-time lane change. In fact, the distribution of the local model at the learners help to take the lane change decision in a tight time window.
- The drone energy consumption is quite acceptable and will encourage road stakeholders to integrate UAVs in the vehicular networks due to the high drone autonomy. It is to be noted that the majority of the energy consumption is related to the drone mobility as compared to the computation and communication.

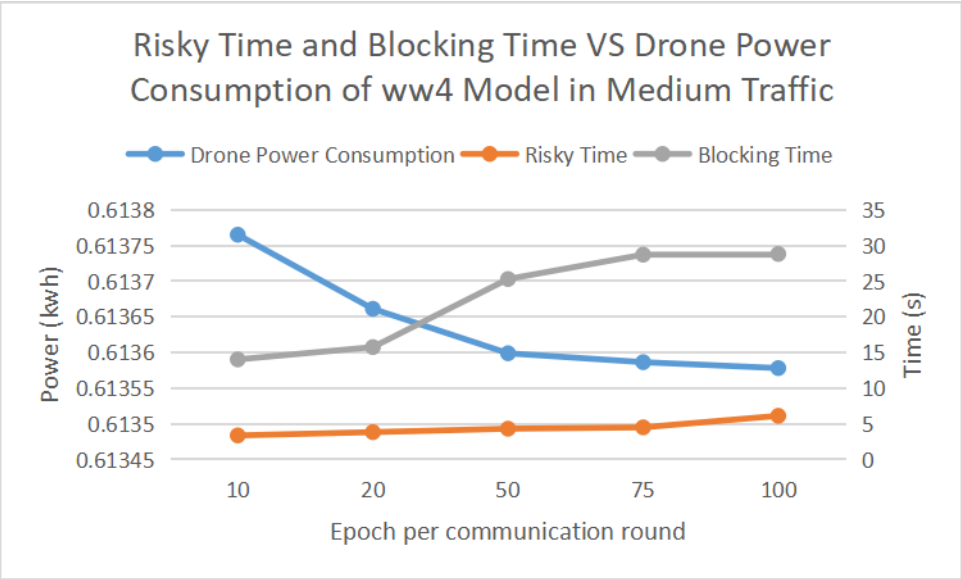

Figure 5.26: Safety performance (risky time and blocking time) vs drone power consumption of ww4 model in medium traffic

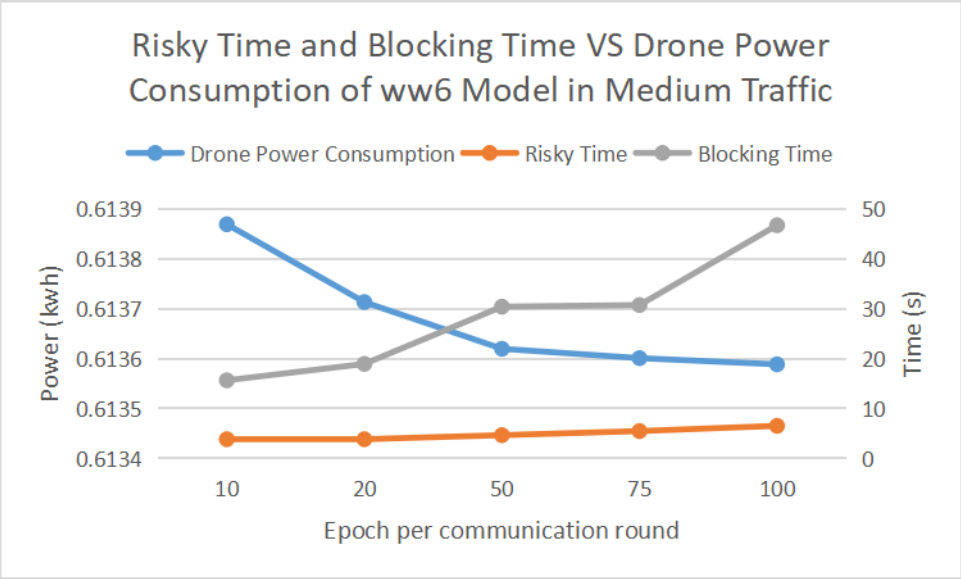

Figure 5.27: Safety performance (risky time and blocking time) vs drone power consumption of ww6 model in medium traffic

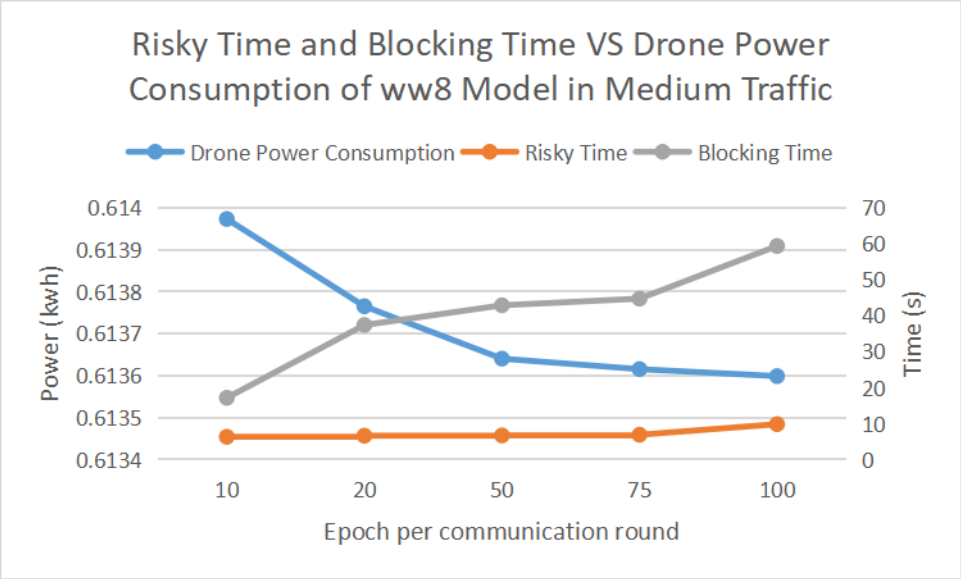

Figure 5.28: Safety performance (risky time and blocking time) vs drone power consumption of ww8 model in medium traffic

5.6 Conclusion

This chapter presents our proposed DAFL drone assisted federated deep reinforcement learning framework. This framework enables the drivers to achieve safe and real-time lane changes. A special concern is dedicated to the computation of the communication delay and drone power consumption which has been proven to be bounded.

In more detail, we first propose a global model aggregation algorithm based on client's reputation for the FL server. Notably, the client's reputation is computed from its local training reward and the cosine similarity between its local update and the global update.

Then, we perform a detailed mathematical analysis of the E2E delay at the drone consisting of the queuing delay and the D2V propagation delay. The queuing delay is formed based on M/G/1 multi-class preemptive queuing model. Moreover, we provide an accurate modeling of the total power consumption of the drone, including computation, communication and mobility.

Afterwards, we propose a dynamic adjustment threshold for the FL global update frequency. The goal is to achieve the best trade-off between safety and power consumption. Consequently, the drone server can dynamically adapt the global model update frequency according to the road safety and its battery life.

Finally, the framework is tested with in-depth simulations. With the simulation results, we are able to determine the adjustment threshold for the FL.

In the next chapter, we further tackle the drone optimal trajectory planning problem in order to enhance the trade-off between drone energy consumption and road safety (i.e. V2D E2E delay).

Chapter 6

Proposed Drone Optimal Placement Algorithm

6.1 Introduction

In this chapter, we propose an optimal drone trajectory prediction algorithm based on ORL in order to reduce the drone energy consumption and V2D E2E delay in the LCA platform. It should be noted that this research work is done with master student Ali Zibara for his 6-month master internship.

In fact, the objective of ORL is to acquire a policy from a static dataset without any additional interactions with the environment. This approach is becoming increasingly crucial for practical applications of reinforcement learning, particularly in fields like robotics, where data collection is time-consuming and potentially risky. Traditional off-policy algorithms tend to struggle when working with fixed datasets, mainly because they make extrapolation errors when dealing with actions that fall outside the dataset's known distribution. This challenge underscores the need to restrict the policy during training to select actions that are within the dataset's known action space. In this case, the Policy in the Latent Action Space (PLAS) method, offers a straightforward way to address this issue, ensuring that this requirement is inherently met [96].

It should be noticed that, in order to train the ORL model, a large dataset is pre-collected. The data features consists of drones' GPS positions, drone energy consumption, and vehicle-to-drone E2E delay, the last two have been detailed in section 5.3 and 5.4. Then, the offline reinforcement model is trained and evaluated in real time simulations.

The rest of this chapter is organised as followed: section 6.2 presents the state-of-the-art of the drone optimal placement algorithms. Section 6.3 and 6.4 explain the E2E delay analysis and the drone energy consumption modeling. Specifically, we highlight the difference between the model in this chapter and in chapter 5. Then, we detail the generation details of the two UAV trajectories: elliptical trajectory and random walk trajectory in section 6.5.2 and 6.5.3. Section 6.6 presents the proposed ORL algorithm with the performance evaluation results. Section 6.6.6 introduce another trajectory prediction approach based on a 3rd-degree polynomial. Finally, section 6.7 ends this chapter with a conclusion.

6.2 State-of-the-Art of Drone Optimal Placement Algorithm

In order to assist the driver in making the optimal lane change decision while optimize battery life, it is necessary to design an optimal trajectory prediction algorithm for the drone. Nevertheless, UAV path planning is a challenging problem that involves finding an efficient and collision-free trajectory for a UAV in a complex and dynamic environment. Many methods have been proposed to solve this problem, such as classical methods, heuristics, meta-heuristics, machine learning, and hybrid algorithms [97]. Each method has its advantages and limitations, depending on the objectives, constraints, and environment of the path planning problem.

6.2.1 Classical methods

Classical methods for UAV path planning involve various techniques to determine a feasible and optimal trajectory for a UAV to navigate through a given environment while avoiding obstacles and reaching a predefined goal [98]. Here's an overview of the three classical methods for UAV path planning:

Grid-based methods

Grid-based methods discretize the environment into a grid or a set of cells. Each cell is labelled as either free (no obstacle) or blocked (occupied by an obstacle). The UAV's path is then planned by searching through this grid. Common algorithms used in grid-based methods include:

- Dijkstra's algorithm: This algorithm computes the shortest path from the UAV's initial position to the goal by expanding nodes in a breadth-first manner. It can be adapted for grid-based path planning by assigning costs to each cell [99].
- A* algorithm: A* algorithm is an extension of Dijkstra's algorithm that incorporates heuristics to guide the search towards the goal more efficiently [100]. It considers both the cost to reach a cell and an estimate of the cost to reach the goal from that cell.

Graph-based methods

Graph-based methods represent the environment as a graph, where nodes represent positions or states, and edges represent possible transitions between states. The UAV's path is then planned by searching through this graph. Common graph-based algorithms include:

- Visibility graphs: This method constructs a graph by connecting visible points in the environment [101]. It reduces the problem of finding an obstacle-free path to finding a path on this graph
- Probabilistic roadmaps (PRMs): PRMs randomly sample the environment to create a graph of connected nodes. Shortest paths can be found by using graph search algorithms on the PRM [102].

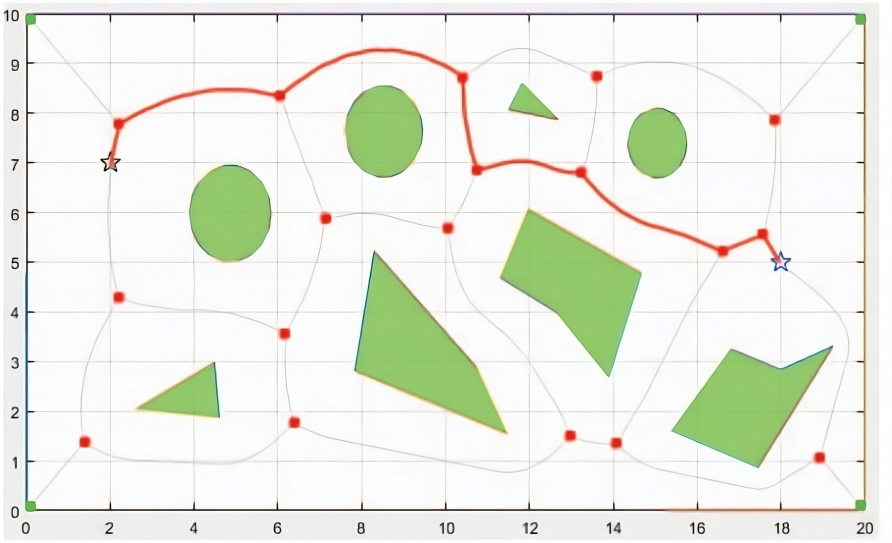

Figure 6.1: Voronoi diagram with the indicated graph branch nodes and optimal point-connected trajectory [104]

Geometric methods

Geometric methods often focus on capturing the geometry of the environment and obstacles to plan UAV paths. Some techniques in this category include:

- Voronoi diagrams: Voronoi diagrams partition space based on the closest obstacle [103, 104]. UAV paths can be planned along the edges of these partitions to ensure clearance from obstacles, as illustrated in Fig. 6.1.
- Potential fields: This method models the environment as a field of attractive and repulsive forces [105]. The UAV is guided towards the goal while being repelled from obstacles.

6.2.2 Optimal control methods

Optimal control methods seek to find the optimal trajectory by considering the dynamic constraints of the UAV and the objective function, which could be minimizing travel time, energy consumption, or another criterion. Common optimal control techniques include:

- Dynamic programming: Dynamic programming breaks down the path planning problem into smaller sub-problems and uses a recursive approach to find the optimal path [106].
- Model Predictive Control (MPC): MPC predicts the future trajectory of the UAV and optimizes control inputs over a finite time horizon to minimize a cost function [107].
- Linear Quadratic Regulator (LQR): LQR is a control technique that optimally controls linear dynamic systems by minimizing a quadratic cost function [108].

Each of these classical methods has its strengths and weaknesses, depending on the specific scenario, environment complexity, UAV dynamics, and planning objectives. Researchers often choose the most appropriate method based on the problem's requirements and constraints.

6.2.3 Heuristic methods

Heuristic methods for UAV path planning leverage various rule-based or learning-based techniques to navigate through an environment while avoiding obstacles and reaching a goal. These methods use heuristics, which are simplified strategies or rules, to guide the UAV's trajectory. We present after an overview of the heuristic methods:

Potential field methods

Potential field methods are based on modeling the environment as a field of attractive and repulsive forces. A UAV is treated as a point mass that moves under the influence of these forces [109]. The goal generates an attractive force, while obstacles create repulsive forces. The UAV follows the gradient of the combined potential field to reach the goal while avoiding obstacles.

- Advantages: Simple to implement, can handle dynamic environments, and can generate smooth trajectories.
- Challenges: Local minima issues (are points in the error surface of a NN where the gradient is zero, but the error is not at its lowest possible value) can cause the UAV to get stuck; difficult to achieve global optimal [110].

Artificial potential field methods

Artificial potential field methods extend the basic potential field concept by incorporating additional features such as tuning parameters, obstacle avoidance strategies, and ways to handle local minima. These methods aim to address some of the shortcomings of simple potential field methods [111].

- Advantages: improved obstacle avoidance, greater flexibility in handling different scenarios [110].
- Challenges: Still susceptible to local minima issues, tuning parameters can be challenging.

Fuzzy logic methods

Fuzzy logic methods involve using fuzzy sets and rules to make decisions in a complex and uncertain environment [112]. These methods allow for reasoning under uncertainty and vagueness by using linguistic variables (e.g., "close", "far", "safe" and "dangerous") instead of strict numerical values.

- Advantages: Can handle imprecise and uncertain information, suitable for situations with qualitative descriptions [113].
- Challenges: Designing appropriate fuzzy rules and membership functions can be complex; might not achieve optimal paths.

Neural network methods

Involve training artificial NNs to learn complex mappings from input (environment information) to output (UAV control signals or path). They can be used to approximate optimal paths or control policies based on training data [114].

- Advantages: Can capture intricate relationships and patterns in data, potentially achieving high accuracy [114].
- Challenges: Requires substantial training data, training can be computationally intensive, and NNs might not provide interpretability [115].

It is worth noting that while these heuristic methods can work well in certain scenarios, they may have limitations when facing highly dynamic or complex environments. Hybrid approaches that combine multiple heuristic methods [116], or combine heuristics with classical methods, are also explored to harness the strengths of different techniques. Additionally, the field of UAV path planning continues to evolve, and researchers are increasingly integrating machine learning and AI techniques to improve the adaptability and performance of heuristic methods.

6.2.4 Meta-heuristic methods

Meta-heuristic methods for UAV path planning are advanced optimization techniques that draw inspiration from various natural phenomena [116], such as evolution, swarm behaviour, and physical principles. These methods are designed to explore complex solution spaces and find near-optimal or optimal paths for UAVs in challenging environments. Here's an overview of some meta-heuristic methods:

Evolutionary algorithms

Evolutionary algorithms are inspired by the process of natural evolution. They operate on a population of potential solutions (individuals), applying selection, crossover (recombination), and mutation operations to iteratively generate better solutions. Genetic algorithms (GAs) and genetic programming (GP) are common evolutionary techniques applied to UAV path planning [117].

- Advantages: Can handle non-convex and multi-modal optimization problems, suitable for complex and large solution spaces.
- Challenges: Convergence to a good solution may be slow; parameter tuning can be crucial.

Swarm intelligence algorithms

Swarm intelligence algorithms are inspired by the collective behaviours of social organisms like ants, bees, and birds. These algorithms involve multiple agents (particles or individuals) interacting with each other and their environment to collectively find solutions [118]. Particle Swarm Optimization (PSO) is a popular example in this category.

- Advantages: capable of exploring the solution space efficiently, good for problems with high-dimensional spaces or multiple objectives
- Challenges: sensitivity to parameter settings, potential for premature convergence.

Physics-based algorithms

Physics-based algorithms simulate physical processes, often inspired by real-world phenomena such as gravity, electromagnetism, and fluid dynamics. They can be applied to optimize UAV paths by treating the problem as a dynamic system where the UAV follows laws of motion [116].

- Advantages: Can produce physically feasible trajectories, suitable for scenarios where dynamics are critical.
- Challenges: Complex modelling of physical interactions, convergence may be affected by simulation accuracy

Other nature-inspired algorithms

There are various other nature-inspired algorithms that draw inspiration from diverse sources, these algorithms often offer innovative ways to explore and exploit solution spaces efficiently:

- Firefly algorithm: Mimics the flashing behaviour of fireflies to optimize solutions in a search space [119].
- Bat algorithm: Inspired by the echolocation and hunting behaviour of bats to optimize paths [120].
- Cuckoo search: Emulates the behaviour of cuckoo birds in searching for hosts' nests [121] as illustrated in Fig. 6.2.
- Artificial bee colony: Simulates the foraging behaviour of honeybees in searching for food sources [122] as illustrated in Fig. 6.3.
- Whale optimization algorithm: Inspired by the social behaviours of humpback whales during bubble-net feeding [123] as illustrated in Fig. 6.4.

Meta-heuristic methods provide a way to tackle complex optimization problems that classical methods might struggle with. However, they often require careful parameter tuning and might not guarantee finding the global optimum. Researchers often experiment with different meta-heuristic methods, hybridize them, and adapt them to specific UAV path planning scenarios to achieve better performance.

6.2.5 Machine learning methods

Machine learning methods have gained significant attention in the field of UAV path planning due to their ability to learn complex behaviours and adapt to various environments. Here's an overview of some machine learning methods applied to UAV path planning:

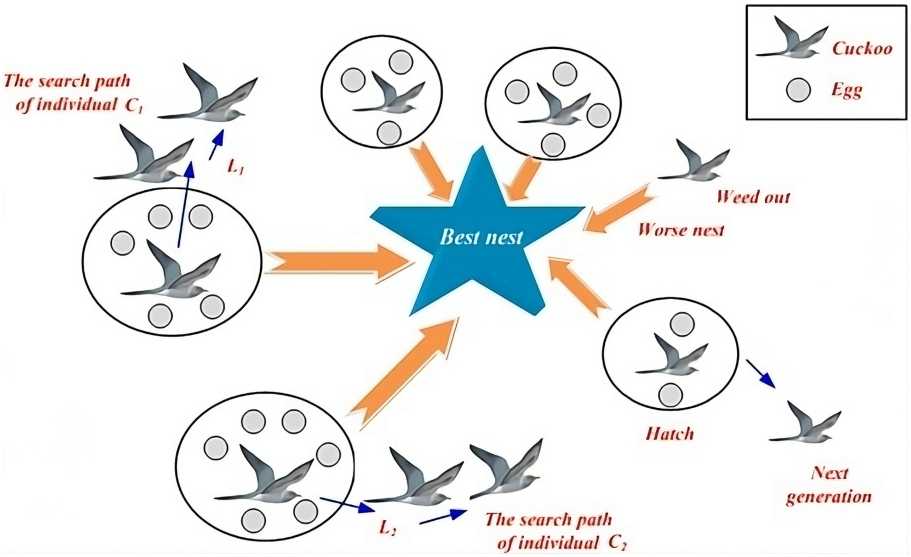

Figure 6.2: Cuckoo search algorithm [121]

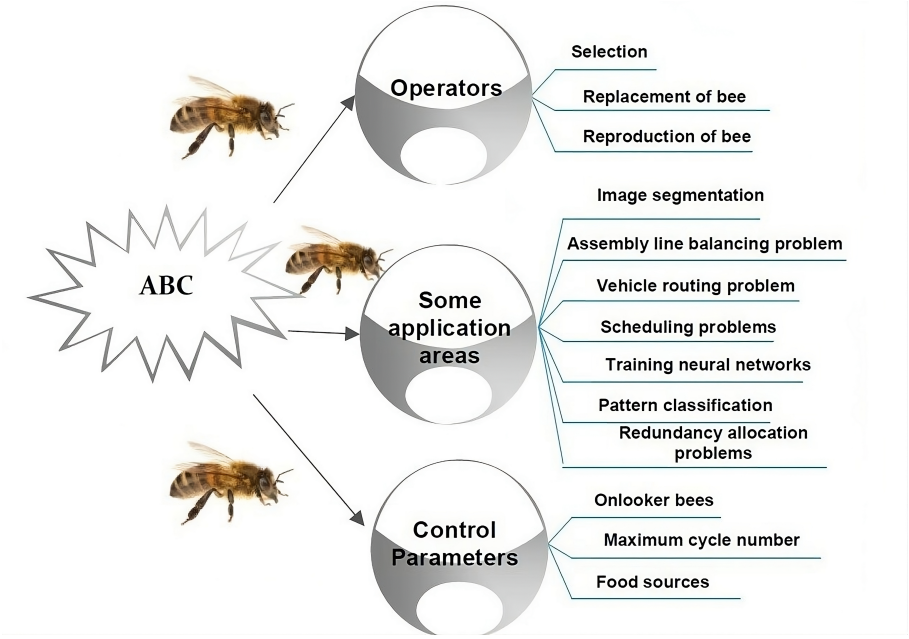

Figure 6.3: Artificial bee colony algorithm [122]

Figure 6.4: Whale optimization algorithm [123]

Reinforcement learning (RL)

RL involves training an agent (UAV) to take actions in an environment to maximize a cumulative reward signal. The agent learns through trial and error, exploring different actions and observing their outcomes [124]. RL algorithms include Q-learning, DQN, and more advanced methods like Trust Region Policy Optimization (TRPO) and Proximal Policy Optimization (PPO).

- Advantages: Can learn optimal policies in complex environments without explicit knowledge of the environment dynamics.
- Challenges: Requires significant training time, exploration challenges, and potential safety concerns during learning.

Deep learning

Deep learning involves training NNs with multiple layers to approximate complex functions. In the context of UAV path planning, deep learning techniques can be applied to learn mappings from sensor data to control actions or paths.

- CNNs: CNNs used for processing sensor inputs like images or LiDAR data to extract features relevant for path planning
- Recurrent Neural Network (RNNs): RNNs suitable for sequential decision-making in dynamic environments [118].
- DRL: DRL combines deep learning and RL to learn policies directly from raw sensor data [125].
- Imitation learning: Imitation learning involves training an agent by mimicking the behaviour of an expert. This is useful when expert demonstrations of desired UAV trajectories are available.

Behavioural cloning

Behavioural cloning trains a model to replicate expert actions based on demonstration data [126]

- Inverse Reinforcement Learning (IRL): IRL infers the underlying reward function that explains the expert's behaviour and then optimizes UAV trajectories based on this learned reward function. It addresses the problem of learning a reward function from observed behaviour [127]. This can be applied to UAV path planning by inferring the preferences or goals of an expert from their trajectories.
  - Advantages: Can capture human intent and preferences, useful for situations where explicit reward functions are hard to define
  - Challenges: Requires careful modelling of the expert's behaviour, can be computationally intensive
- Imitation learning: Imitation learning can help generate safe and reliable paths based on demonstrated human expertise.

Offline reinforcement learning (ORL)

ORL for UAV path planning involves training a RL agent in an environment where interactions with the actual UAV are not required during the training phase [128, 129]. Instead, the agent learns from a per-collected dataset of state-action pairs that simulate UAV behaviour in various scenarios as illustrated in Fig. 6.5 and Fig. 6.6. This approach is particularly useful when direct online interaction with the real UAV might be costly, time-consuming, energy-consuming or potentially risky.

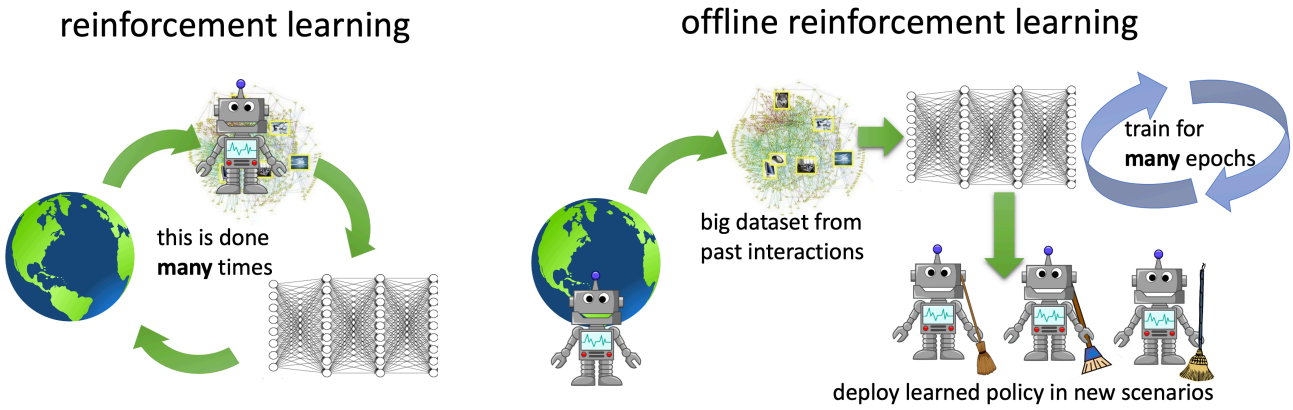

Figure 6.5: Reinforcement learning Vs. offline reinforcement learning [129]

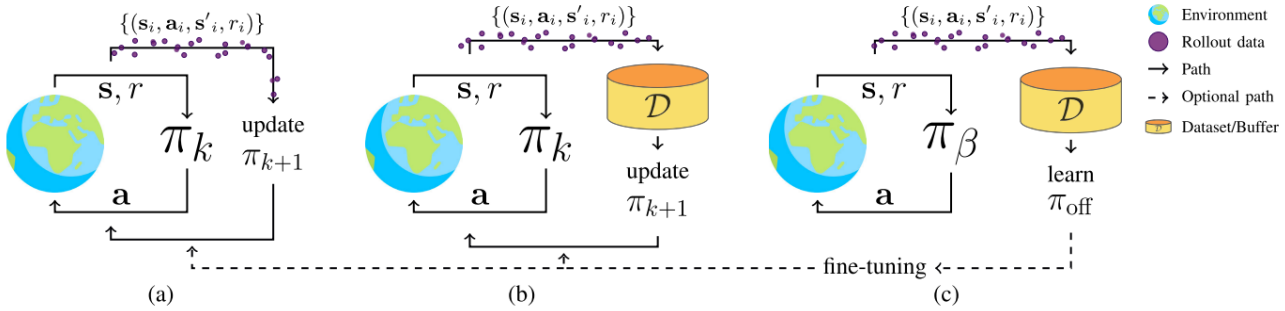

Figure 6.6: Illustration of the different RL paradigms, including (a) online RL, (b) off-policy RL, and (c) ORL. In online RL new experiences must becollected with the latest policy before making an update. In off-policy RL, we reuse previous experiences but still rely on a continuous collection of new experiences. In contrast, ORL only uses previous experiences collected with a behavior policy π_β and stored in a static dataset D to learn a policy π_{off} .After learning off, one can opt to fine-tune it using online or off-policy RL methods [130]

A high-level overview of the process of ORL for UAV path planning is listed below:

1. Data collection: In the ORL setting, a dataset of state-action pairs is collected beforehand. Each pair consists of the observed state of the environment (such as UAV's position, velocity, sensor data, etc.) and the corresponding action taken by the expert or an existing controller. This data can come from expert demonstrations, historical flight data, or simulations

2. Algorithm selection: This process chooses an ORL algorithm suitable for the problem. Common algorithms include Batch Q-learning, Batch Constrained Q-learning, and Conservative Q-learning. These algorithms are designed to work with fixed datasets and learn policies based on the collected data.

3. Reward function: This process is to define the reward function that reflects the goals and objectives of the UAV path planning problem. This can involve shaping the rewards to guide the agent toward desired behaviours, such as obstacle avoidance, reaching a goal, or energy efficiency.

4. Policy learning: This process aims to use the collected dataset and the chosen RL algorithm to train a policy that maps states to actions [130]. The algorithm will learn a policy that aims to maximize the cumulative reward over the dataset trajectories.

5. Evaluation and fine-tuning: After training, evaluate the learned policy using metrics relevant to the UAV path planning problem. If the performance is not satisfactory, we might need to fine-tune hyper-parameters, modify the reward function, or retrain the policy with additional data.

Benefits of ORL for UAV path planning:

- Safety: Since the learning process occurs offline, there is no risk of causing harm to the actual UAV during training.
- Efficiency: Training can be computationally intensive, and ORL avoids the need for real-time interactions.
- Expert guidance: ORL can incorporate expert behaviour, leveraging human knowledge to guide the learning process.
- Data re-usability: Once trained, the learned policy can be reused in various scenarios without retraining.

Challenges of ORL for UAV path planning:

- Data quality: The quality and diversity of the training dataset can significantly impact the learned policy's performance.
- Generalization: The learned policy might not generalize well to unseen environments or conditions.
- Sample efficiency: ORL methods often require large datasets for effective learning.

Machine learning methods offer the advantage of adaptability and learning from data, enabling UAVs to navigate in complex and dynamic environments. However, these methods often require substantial amounts of data, significant computational resources, and careful consideration of safety and generalization to new scenarios. Researchers are actively exploring ways to combine machine learning techniques with traditional methods to achieve robust and efficient UAV path planning solutions. It's important to carefully design the experiment, choose the appropriate

algorithm, and consider the practical implications of applying ORL to UAV path planning, such as the potential for over-fitting to the training data and limitations in adapting to new situations.

6.2.6 Hybrid algorithms

Hybrid algorithms for UAV path planning combine two or more different methods from various categories, such as classical methods, heuristic methods, meta-heuristic methods, and machine learning methods [131]. These combinations are aimed at leveraging the strengths of each individual method to achieve better performance, robustness, and adaptability in challenging UAV path planning scenarios. Here are some common types of hybrid algorithms:

Classical-heuristic hybrids

These hybrids combine classical methods with heuristic approaches to balance computational efficiency with improved obstacle avoidance or path smoothness. For example, combining a grid-based A* search with potential field methods can enable the UAV to navigate complex environments while maintaining a smooth trajectory.

Heuristic-meta-heuristic hybrids

These hybrids integrate heuristic methods with meta-heuristic techniques to enhance exploration and exploitation of the solution space. For instance, combining particle swarm optimization (meta-heuristic) with artificial potential fields (heuristic) can improve the ability to find optimal paths while avoiding local minima.

Machine learning-heuristic hybrids

These hybrids use machine learning techniques to fine-tune or guide heuristic methods. For instance, a machine learning model could predict the effectiveness of a potential field in different regions of the environment, allowing the heuristic algorithm to adjust its behaviour accordingly.

Machine learning-meta-heuristic hybrids

These hybrids incorporate machine learning methods with meta-heuristic approaches to improve efficiency and solution quality. A NN could be used to guide the exploration of a particle swarm optimization algorithm, providing more informed search directions.

Adaptive hybrids

Dynamically switch between different methods based on the current state of the environment or the UAV. For example, a UAV might use classical methods for path planning in known areas and switch to a RL-based approach in unfamiliar or changing environments.

Multi-objective hybrids

In multi-objective path planning scenarios, hybrid algorithms can combine methods to optimize conflicting objectives simultaneously. This might involve combining a genetic algorithm with a Pareto-based optimization technique to find a trade-off between objectives like minimizing path length and maximizing safety.

Benefits of hybrid algorithms for UAV path planning

- Improved performance: By combining complementary methods, hybrid algorithms can achieve better performance than using individual methods alone.
- Robustness: Hybrids can handle a wider range of scenarios, including complex and dynamic environments, by adapting to different challenges.
- Solution quality: Combining methods can result in higher-quality paths, overcoming limitations of individual methods.

Challenges of hybrid algorithms for UAV path planning

- Complexity: Designing effective hybrid algorithms requires careful consideration of method integration, parameter tuning, and possible interactions.
- Trade-offs: Combining methods may introduce trade-offs between competing goals, such as optimization speed versus solution quality.
- Generalization: Ensuring that hybrid algorithms generalize well to various scenarios can be challenging.

Hybrid algorithms are a flexible approach to addressing the complexities of UAV path planning and are often tailored to specific application requirements. They are an active area of research, with efforts focused on designing and testing novel combinations to achieve optimal performance and adaptability.

6.2.7 Conclusion

In the previous sections, we have surveyed the state-of-the-art methods and challenges for UAV path planning, which is the problem of finding optimal and collision-free trajectories for UAVs in complex and dynamic environments [114]. We have classified the existing methods into five main categories: classical methods, heuristics, meta-heuristics, machine learning, and hybrid algorithms. The comparison between different methods are provided in Table. 6.1.

Furthermore, we have also analysed the advantages and limitations of each category based on the objectives, constraints, and environments of the path planning problem. Furthermore, we have identified some future research directions for enhancing the performance and robustness of UAV path planning methods. This section can provide a comprehensive and up-to-date overview of the UAV path planning field and inspire further research and development in this

area. In the next sections, the motivation to choose ORL, the implementation and the results achieved will be discussed in detail.

Table 6.1: Comparison table of UAV path planning methods

Method	Description	Advantages	Disadvantages
Grid-Based	Discretize the environment into a grid or a set of cells. Search through the grid using algorithms like Dijkstra' s or A*.	Simple and easy to implement. Can handle complex environments.	Computationally expensive. Resolution dependent
Graph-Based	Represent the environment as a graph, where nodes represent positions or states, and edges represent possible transitions. Search through the graph using algorithms like Visibility Graphs or PRMs	Flexible and scalable. Can handle high-dimensional spaces.	May not capture all the details of the environment. May require preprocessing.
Geometric	Capture the geometry of the environment and obstacles to plan UAV paths. Use techniques like Voronoi Diagrams or Potential Fields.	Can handle non-convex obstacles. Can ensure safe distances from obstacles	May not be optimal. May have local minima.
Optimal Control	Find the optimal trajectory by considering the dynamic constraints of the UAV and the objective function. Use techniques like Dynamic Programming, MPC, or LQR.	Can handle complex dynamics and objectives. Can guarantee optimality and stability	Computationally intensive. May require accurate models and measurements.
Potential Field	Model the environment as a field of attractive and repulsive forces. The UAV follows the gradient of the combined potential field to reach the goal while avoiding obstacles.	Simple to implement, can handle dynamic environments, can generate smooth trajectories.	Local minima issues can cause the UAV to get stuck, difficult to achieve global optimality.
Artificial Potential Field	Extend the basic potential field concept by incorporating a dditional features such as tuning parameters, obstacle avoidance strategies, and ways to handle local minima.	Improved obstacle avoidance, greater flexibility in handling different scenarios.	Still susceptible to local minima issues, tuning parameters can be challenging.

Table 6.1 continued from previous page

Method	Description	Advantages	Disadvantages
Fuzzy Logic	Use fuzzy sets and rules to make decisions in a complex and uncertain environment. Use linguistic variables instead of strict numerical values	Can handle imprecise and uncertain information, suitable for situations with qualitative descriptions.	Designing appropriate fuzzy rules and membership functions can be complex; might not achieve optimal paths.
Neural Network	Train artificial neural networks to learn complex mappings from input (environment information) to output (UAV control signals or path). Approximate optimal paths or control policies based on training data.	Can capture intricate relationships and patterns in data, potentially achieving high accuracy	Requires substantial training data, training can be computationally intensive, and NNs might not provide interpretability
Evolutionary	Inspired by the process of natural evolution. Operate on a population of potential solutions, applying selection, crossover, and mutation operations to iteratively generate better solutions.	Can handle non-convex and multi modal optimization problems, suitable for complex and large solution spaces.	Convergence to a good solution may be slow; parameter tuning can be crucial.
Swarm Intelligence	Inspired by the collective behaviours of social organisms like ants, bees, and birds. Involve multiple agents interacting with each other and their environment to collectively find solutions.	Capable of exploring the solution space efficiently, good for problems with high-dimensional spaces or multiple objectives.	Sensitivity to parameter settings, potential for premature convergence
Physics-Based	Simulate physical processes, often inspired by real-world phenomena such as gravity, electromagnetism, and fluid dynamics. Treat the problem as a dynamic system where the UAV follows laws of motion.	Can produce physically feasible trajectories, suitable for scenarios where dynamics are critical.	Complex modelling of physical interactions, convergence may be affected by simulation accuracy.

Table 6.1 continued from previous page

Method	Description	Advantages	Disadvantages
Other Nature-Inspired	Draw inspiration from diverse sources such as fireflies, bats, cuckoos, bees, and whales. Offer innovative ways to explore and exploit solution spaces efficiently.	Can handle complex and dynamic environments, can adapt to different scenarios	Require careful parameter tuning, might not guarantee finding the global optimum.
Reinforcement Learning	Train an agent to take actions in an environment to maximize a cumulative reward signal. The agent learnd through trial and error, exploring different actions and observing their outcomes.	Can learn optimal policies in complex environments without explicit knowledge of the environment dynamics.	Requires significant training time, exploration challenges, and potential safety concerns during learning.
Deep Learning	Train NNs with multiple layers to approximate complex functions. Learn mappings from sensor data to control actions or paths.	Can capture intricate relationships and patterns in data, potentially achieving high accuracy and adaptability.	Requires substantial training data, training can be computationally intensive, and neural networks might not provide interpretability
Behavioural Cloning	Train a model to replicate expert actions based on demonstration data.	Can help generate safe and reliable paths based on demonstrated human expertise	Might not generalize well to unseen situations, might suffer from compounding errors
Inverse Reinforcement Learning	Learn a reward function from observed behaviour. Infer the preferences or goals of an expert from their trajectories	Can capture human intent and preferences, useful for situations where explicit reward functions are hard to define.	Requires careful modelling of the expert' s behaviour, can be computationally intensive
Offline Reinforcement Learning	Train a reinforcement learning agent in an environment where interactions with the actual UAV are not required during the training phase. Use a pre collected dataset of state-action pairs that simulate UAV behaviour in various scenarios.	No risk of causing harm to the actual UAV during training, avoids the need for real-time interactions, incorporates expert behaviour, reuses learned policy in various scenarios.	Requires large and diverse datasets, might not generalize well to unseen environments or conditions, training can be computationally intensive.

6.3 End to End Delay Analysis

This section analyzes the V2D communication delay with the help of queuing theory, as illustrated in Fig. 6.7. It should be noticed that in this section, the total V2D delay of a class i request from a vehicle j, denoted as $E[W_{i,j}]$, consists of three parts, which is different from the scenario presented in Fig. 5.5:

- V2D propagation delay, $W_{i,j}^{V2D}$,
- queuing delay at the drone side, $W_{i,j}^{s}$,
- and drone-to-vehicle (D2V) propagation delay, $W_{i,j}^{D2V}$.

Thus, instead of (5.5), now we have

$$E[W_{i,j}] = E[W_{i,j}^{V2D}] + E[W_{i,j}^{s}] + E[W_{i,j}^{D2V}] \leq T_i, \forall j \in \mathbb{V}, \tag{6.1}$$

where the parameters have the same physical meanings as (5.5).

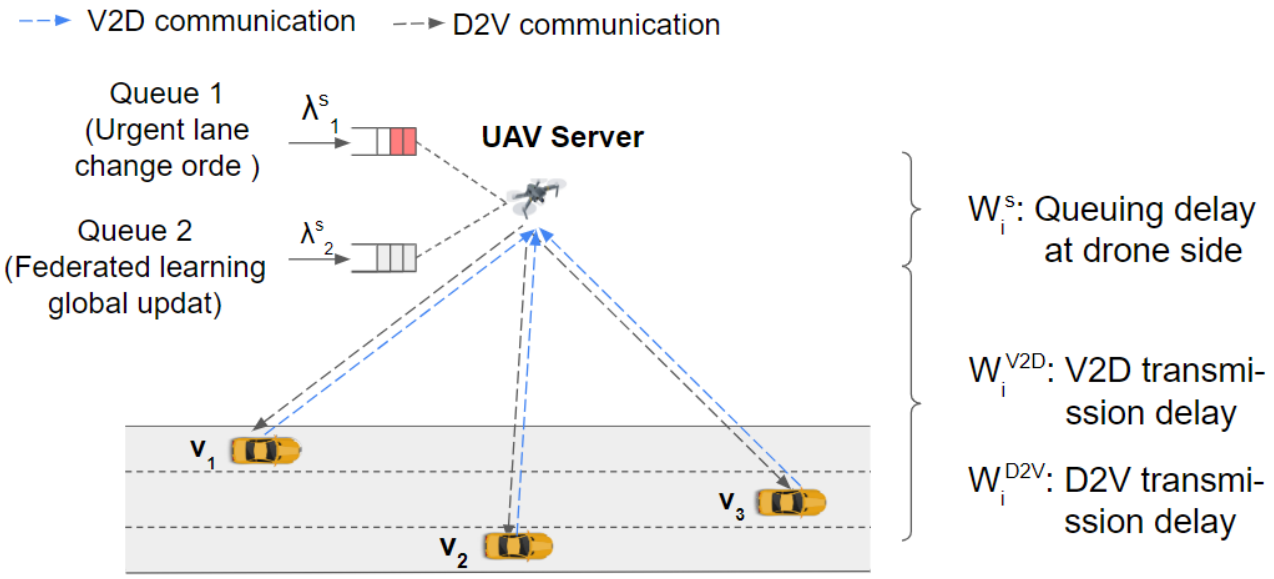

Figure 6.7: Illustration of the E2E delay analysis with both D2V and V2D communications in DAVN.

In fact, the total delay of a class i request is (5.5) plus the V2D propagation delay. As we have

$$E[W_{i,j}^{V2D}] = \frac{d'_{i,j}}{r_i}, \tag{6.2}$$

where $d'_{i,j}$ is the distances between the UAV and the vehicle j from which the request is sent during the V2D communication, the total delay can be calculated as

$$E[W_{i,j}] = \begin{cases} \frac{d_{i,j}}{r_i} + \frac{d'_{i,j}}{r_i} + \frac{\rho_1}{2(1-\rho_1)} \cdot \frac{E[B_1^2]}{E[B_1]} + E[B_1], & \text{for class 1 requests} \\ \frac{d_{i,j}}{r_i} + \frac{d'_{i,j}}{r_i} + \frac{1}{(1-(\rho_1+\rho_2))(1-\rho_1)} \cdot \sum_{j=1}^{2} \rho_j \frac{E[B_i^2]}{2E[B_i]} + E[B_2], & \text{for class 2 requests} \end{cases} \tag{6.3}$$

6.4 Energy Consumption Model

In this section, the drone energy consumption of UAV i consists of three parts: the communication energy that is used for the data transfer from the UAV i to vehicle j, $E_{i,j}^{D2V}$, and to other UAVs, $E_{i,j}^{D2D}$, and the propulsion energy of the UAV to adjust its location for data transfer, E_i^m, as in the following equations:

$$E_i = \sum_{j \in \mathbb{V}} E_{i,j}^{D2V} + \sum_{m \in \mathbb{U}} E_{i,m}^{D2D} + E^m, \tag{6.4}$$

where $\mathbb{V}$ is the set of all vehicles in the communication range of the UAV i, $\mathbb{U}$ denotes the set of all UAVs. One can see that we consider both the D2V and the Drone-to-Drone (D2D) communications. In the following sections, each of the components will be analysed. With the same analysis as in section 5.4, we can derive the D2D communication energy as in the following paragraph:

Drone-to-drone path loss The D2D path loss is also known as air-to-air path loss. Contrary to the D2V scenario, the D2D path loss is dominated by the free-space LoS propagation. Thus the D2D LoS path loss between UAV n and UAV m is given as the same as the D2V communication, as described in Equation. (6.5) [132]:

$$PL_{n,m}^{D2D} = 20\log_{10}\left(\frac{4\pi f_c d_{n,m}^{euc}}{c}\right) + \eta_{LoS}^{D2D}, \tag{6.5}$$

where η_{LoS}^{D2D} is the mean additional loss of the D2D communication link, c is the speed of light, and $d_{n,m}^{euc} = \sqrt{(x_n - x_m)^2 + (y_n - y_m)^2}$ is the euclidean distance between UAV n and UAV m. Consequently, the channel gain $G_{n,m}^{D2D}$, $\text{SNR}_{n,m}^{D2D}$, achievable data rate $C_{n,m}^{D2D}$ and energy consumption $E_{n,m}^{D2D}$ for transmitting a message of size S_n are calculated as followed:

$$G_{n,m}^{D2D} = \frac{1}{PL_{n,m}^{D2D}} \tag{6.6}$$

$$\text{SNR}_{n,m}^{D2D} = \frac{p_n G_{n,m}^{D2D}}{\sum_{i \in \mathbb{N}_{\text{int}}} p_i G_{i,j}^{D2D} + N_0} \tag{6.7}$$

$$C_{n,m}^{D2D} = B * \log_2\left(1 + \text{SNR}_{n,m}^{D2D}\right) \tag{6.8}$$

$$E_{n,m}^{D2D} = \frac{S_n}{C_{n,m}^{D2D}} \, p_n^{trans} \tag{6.9}$$

As for the mobility energy model, it is the same as in section 5.4. Thus, the total drone energy consumption can be modeled as (6.10):

$$\begin{aligned} E_i &= \sum_{j\in\mathbb{V}} E_{i,j}^{D2V} + \sum_{m\in\mathbb{U}} E_{i,m}^{D2D} + E^m \\ &= \sum_{j\in\mathbb{V}} \frac{S_i}{C_{i,j}^{D2V}}\ p_i^{trans} + \sum_{m\in\mathbb{U}} \frac{S_n}{C_{n,m}^{D2D}}\ p_n^{trans} + \frac{\sqrt{(x_1-x_2)^2+(y_1-y_2)^2+(z_1-z_2)^2}}{V_i} P_0 \left(1+\frac{3V_i^2}{U_{\text{tip}}^2}\right) \\ &+ P_1 \left(\sqrt{1+\frac{V_i^4}{4v_0^4}} - \frac{V_i^2}{2v_0^2}\right)^{1/2} + \frac{1}{2} d_0 \rho s A V_i^3, \end{aligned} \tag{6.10}$$

where the corresponding parameters are defined in previous sections.

6.5 Dataset Generation

6.5.1 Simulation setup

We consider the same highway scenario as our previous work, illustrated in Fig. 5.4. We adopt two types of trajectories for the drones: elliptical trajectory and random walk trajectory. Both trajectories will be detailed in the following sections. The maximum horizontal speed of UAVs is 30 km/h, the maximum vertical speed is 10 km/h, the limited flying height is between 100 meters and 150 meters [2, 133].

On the other hand, vehicle trajectories are retrieved by the TraCI interface of SUMO [25, 93]. Vehicles move according to the Krauss mobility model and LC2013 lane change model. The maximum allowed velocity is 100 km/h. For a more authentic scenario, some vehicles are set to be "aggressive" with impolite behaviors such as low intention to cooperate with others, stay in the leftmost lane for a long time and exceed the speed limit, represented by the yellow cars. Moreover, some vehicles are ambulances that have higher priority to pass and lead the in front vehicles to initiate lane change, represented by the red cars [72].

6.5.2 Elliptical trajectory

Horizontal trajectory for x and y

We adopt elliptical trajectory for the four UAVs as illustrated in Fig. 6.9. The UAVs move according to the elliptic curve. The initial mathematical expression is shown in Equation. (6.11):

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1, \tag{6.11}$$

where 2a is the width of the ellipse and 2b is the height of the ellipse.

In our simulation scenario, the four ellipses are distributed on a circle centered on $(0, 0)$ with a radius of 637 meters, representing the circular highway, as shown in Fig. 6.8. Consequently, $2a = 637 \times 2 = 1274$, $2b = 637$. The horizontal speed and vertical speed of the drones are

liimited to 30km/h ($\approx$ 8.33m/s) and 10km/h ($\approx$ 2.78m/s). The traffic state is updated every 0.4 second. Thus, x and y positions are functions of simulation step, s, where v_{hor} and v_{vrt} are the horizontal speed and vertical speed in meter per second:

$$x = v_{hor} \times 0.4s$$
$$y = v_{vrt} \times 0.4s$$

It should be noted that the speed of the four drones are set to 5, 10, 20, 30 km/h. On the other hand, the center coordinates of the four UAVs are

$$(Cx_1, Cy_1) = (0, 637)$$
$$(Cx_2, Cy_2) = (637, 0)$$
$$(Cx_3, Cy_3) = (0, -637)$$
$$(Cx_4, Cy_4) = (-637, 0)$$

Finally, the trajectories with expression in the form of

$$\frac{(x - Cx_i)^2}{a^2} + \frac{(y - Cy_i)^2}{b^2} = 1$$

for the four UAVs can be represented as in the following equations:

$$\frac{x^2}{a^2} + \frac{(y - 637)^2}{b^2} = 1$$
$$\frac{(x - 637)^2}{a^2} + \frac{y^2}{b^2} = 1$$
$$\frac{x^2}{a^2} + \frac{(y + 637)^2}{b^2} = 1$$
$$\frac{(x + 637)^2}{a^2} + \frac{y^2}{b^2} = 1$$

Vertical trajectory for z

We adopt a random walk trajectory to determine $z_i, i \in [1, 4]$. It should be noticed that at each step, the UAV moves up or down according to the predefined probability array $p = (p_1, 1 - p_1)$, where p_1 is the probability to moving up, and p_2 is the probability to moving down. In the simulation, we set $p = (0.5, 0.5)$. The random walk algorithm is detailed in Algorithm. 4.

Final trajectories

Examples of the generated trajectories of the UAVs are shown in Fig. 6.10.

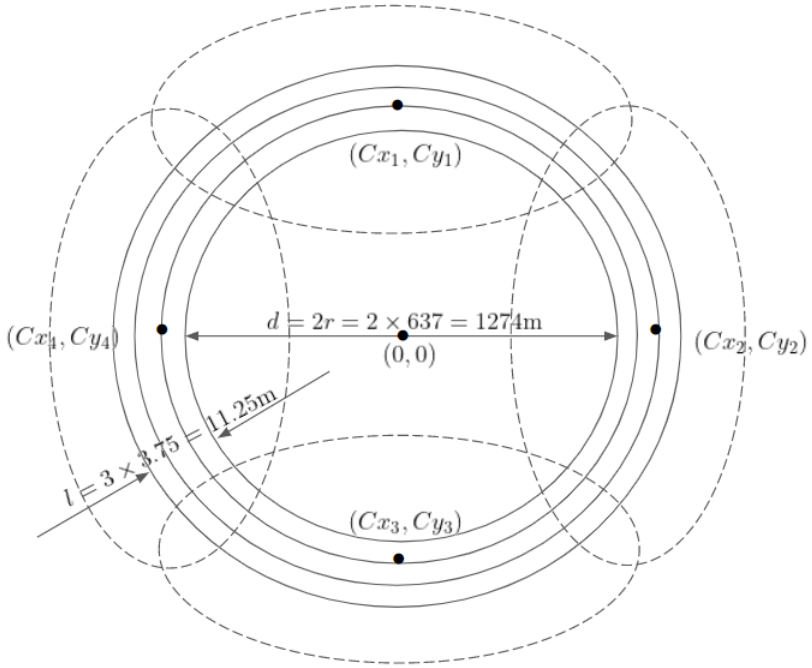

Figure 6.8: Details of the ellipses

Algorithm 4 `Vertical trajectory generation`

Input: S is the number of simulation steps, Z is the vertical action range, p and $1-p$ are the probabilities to go up and down at each step, Δz is the vertical step length
Initialize vertical trajectory as an all-zero array $\boldsymbol{z}$ of length S
for $i \leftarrow 0$ to S **do**
    Randomly choose a vertical direction, 1 for up, -1 for down, according to p
    for the chosen direction j, $j \in \{-1, 1\}$ **do**
        if $\boldsymbol{z}[i] + j * \Delta z \in Z$ **then**
            Perform the movement
        end if
    end for
end for

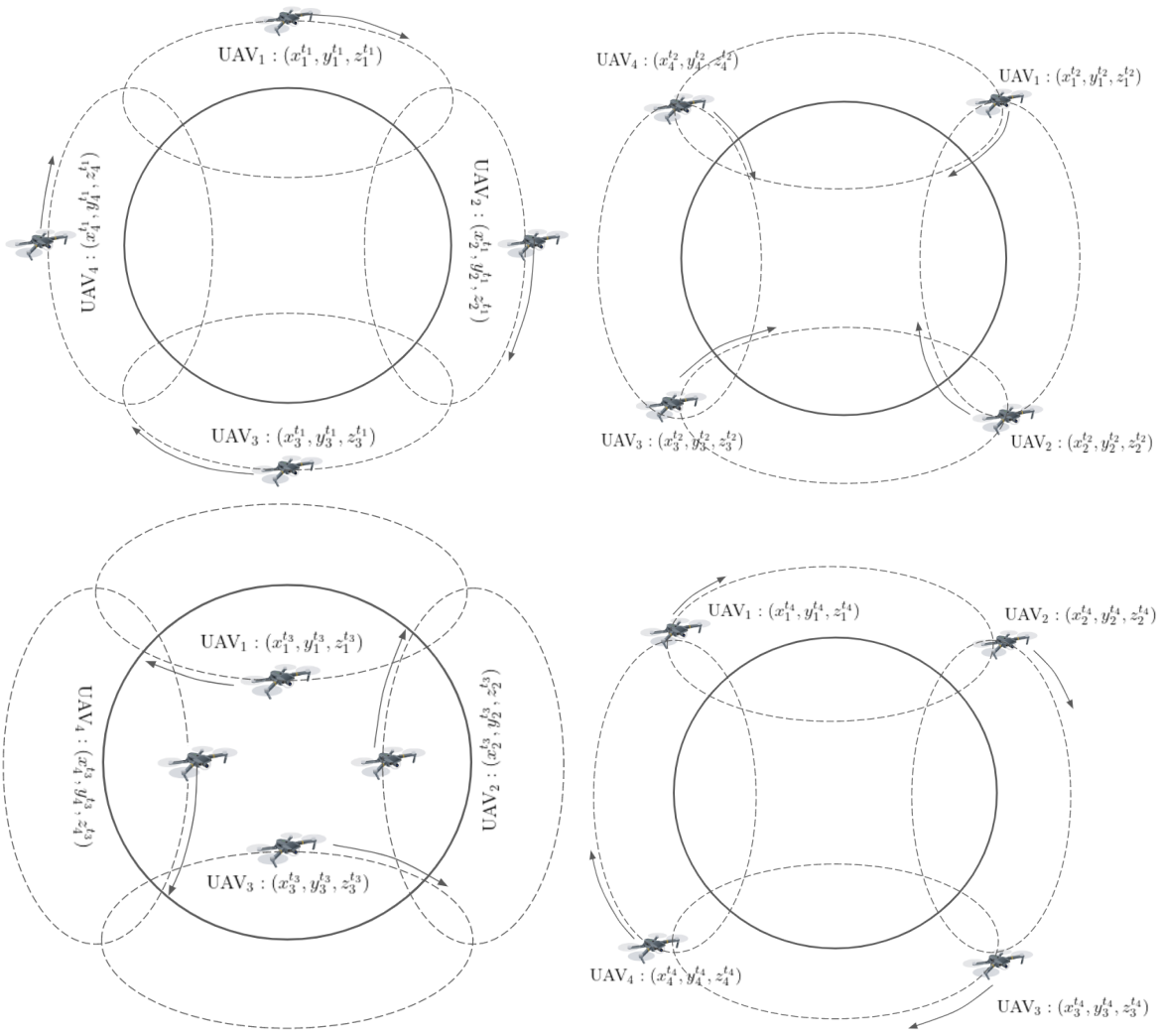

Figure 6.9: Examples of the generated elliptical trajectories of UAVs at time t_1, t_2, t_3 and t_4

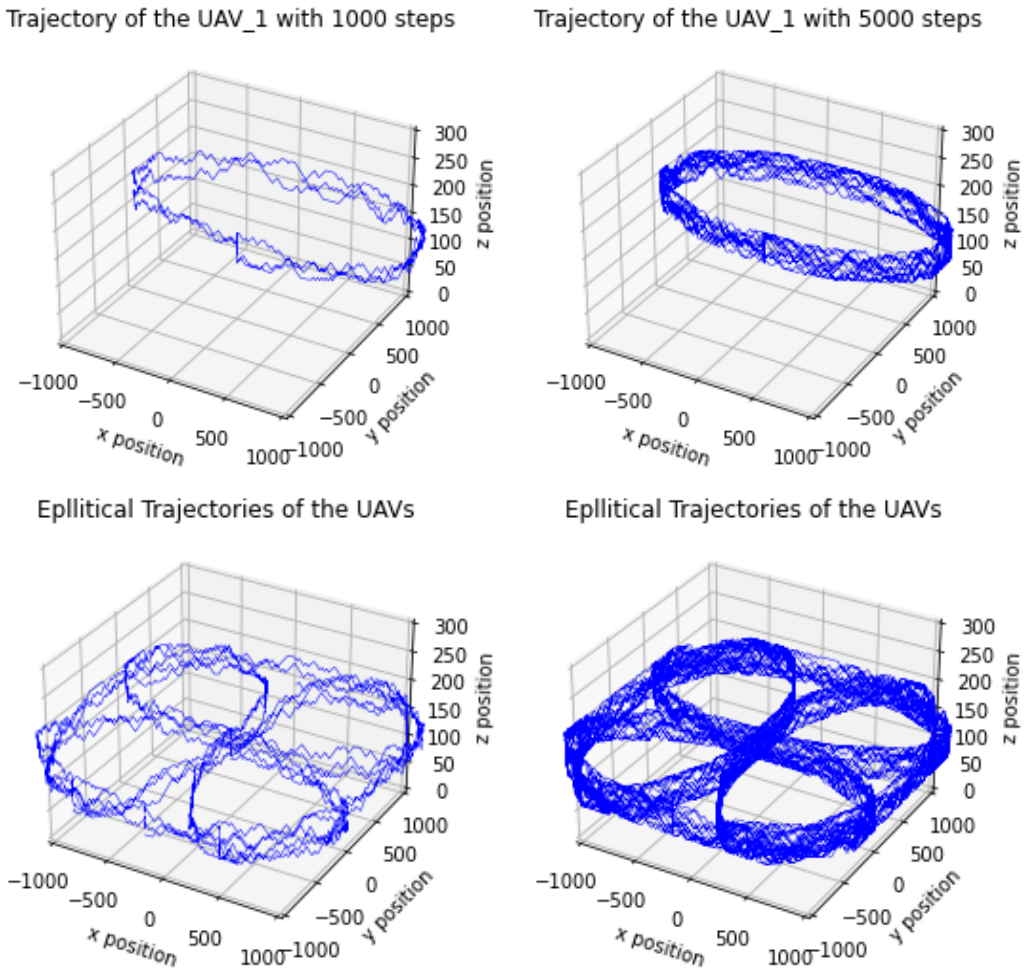

Figure 6.10: Elliptical trajectories of UAVs

6.5.3 Random walk trajectory

Trajectory generation algorithm

In this section, we introduce the random walk trajectories for drones. At each step, the drone move a random step in the sens of random step length and random direction. The maximum speeds for x, y and z are defined as 20km/h, 20km/h and 10 km/h in order to meet the UAV speed constraints. As illustrated in Fig. 5.4, the trajectory of each drone is limited in its communication range. The trajectory generation algorithm is presented in Algo. 5.

Algorithm 5 `Random walk trajectory generation`

Input: S is the number of simulation steps, R_x is the horizontal range on x axis, R_y is the horizontal range on y axis, R_z is the vertical range, Δy, Δz are the step lengths of the three directions, p and $1-p$ are the probabilities to move forward and backward for each direction
for $i \leftarrow 0$ to S **do**
  for each axis $a, a \in \{x, y, z\}$ **do**
    Randomly choose a moving direction $j_a, j_a \in \{-1, 1\}$ according to p
    Randomly choose a step length $l_a, l_a \in [0, \Delta a]$
  end for
  if $a[i] + j_a * l_a \in R_a$ **then** ▷ The moving range constraint
    Perform the movement
  else
    Repeat step 3-8 until moving range constraint is meet
  end if
end for

Final Trajectories

The generated trajectories for the four drones are illustrated in Fig. 6.11.

6.5.4 Dataset parameters

The dataset will be generated through a branch of simulations with different vehicular density ρ. The samples (i.e. the vehicles and drones kinematic parameters) are retrieved, computed and stored every 0.4s. The parameters used for simulation are shown in Table. 6.2. Each sample consists of the observations of each of the 4 UAVs, as well as the overall collision rate on the highway, as represented by the following:

$$o = \{o_1, o_2, o_3, o_4, \bar{r}\} \tag{6.12}$$

where o_1, o_2, o_3, and o_4 are the observations of UAV_1,UAV_2,UAV_3 and UAV_4, respectively. $\bar{r}$ is the total collision number on the highway.
An observation of UAV i at step t is expressed as

$$o_i[t] = \left\{x_i[t], y_i[t], z_i[t], \theta_i[t], \rho_i, \bar{W}_i, \bar{E}_i, \bar{t_{ri}}, \bar{t_{bi}}\right\}, \tag{6.13}$$

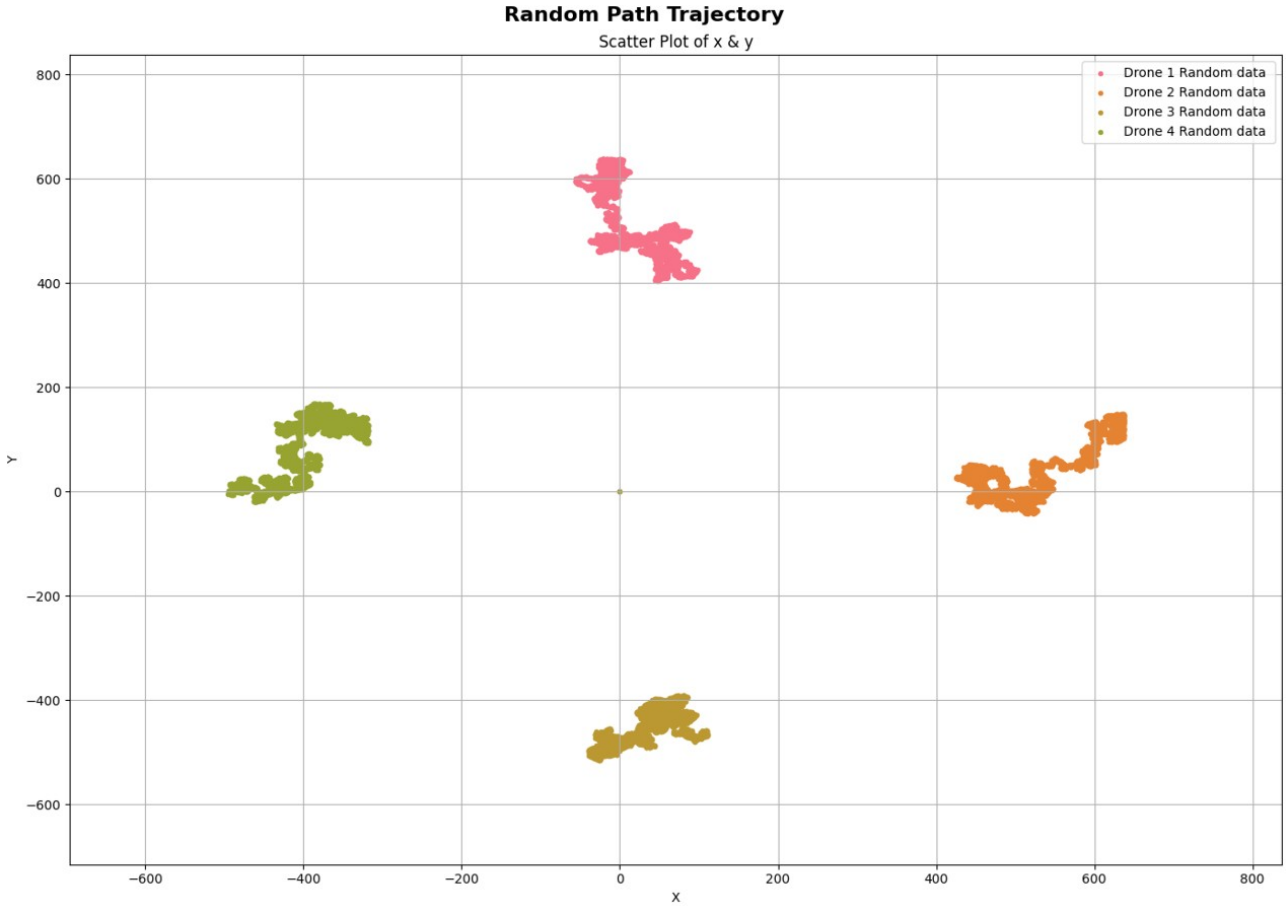

Figure 6.11: Random walk trajectories generated for the four drones

where $x_i[t]$ is the longitudinal position of UAV i at step t, $y_i[t]$ denotes the lateral position at step t, $z_i[t]$ is the height at step t, $\theta_i[t]$ is the heading direction at step t ($\theta_i[t] = 1$ means moving up, $\theta_i[t]$=-1 means moving down), ρ_i is the number of vehiculars in the communication range of UAV i, $\bar{W}_i$ the mean waiting time of the vehicle requests in the communication range of UAV i, $\bar{E}_i$ the average energy consumption of UAV i, $\bar{t_{ri}}$ and $\bar{t_{bi}}$ the mean risky time and mean blocking time of the vehicles in the communication range of the UAV i. The algorithm for generating the dataset is presented in Algorithm. 6

Algorithm 6 `Algorithm for dataset generation`

Input: S is the number of simulation steps, $\mathbb{V}$ is the set of vehicles on the road
Initialize
for $t \leftarrow 0$ to S **do**
    for every vehicle j, $j \in \mathbb{V}$ **do**
        Determine the corresponding UAV of the vehicle according to the vehicle's GPS position and UAV's coverage, denote the corresponding UAV as UAV i
        Compute W_i^j, E_i^j according to Equation. (6.1) and Equation. (6.4)
        Retrieve t_{ri}^j, t_{bi}^j from vehicle state information
        Store W_i^j, E_i^j, t_{ri}^j and t_{bi}^j in the UAV i's buffer
    end for
    for every UAV i, $i \in [1, 4]$ **do**
        Compute the average $\bar{W}_i$, $\bar{E}_i$, $\bar{t_{ri}}$, $\bar{t_{bi}}$
        Store current position $x_i[t]$, $y_i[t]$, $z_i[t]$, current heading direction $\theta_i[t]$, current vehicular density ρ_i, and $\bar{W}_i$, $\bar{E}_i$, $\bar{t_{ri}}$, $\bar{t_{bi}}$ as current observation of UAV i
    end for
    Store $\bar{r}$, number of collisions happen at the current step
end for

6.5.5 Notations and terminologies

The notations and terminologies used in this report are summarized in Table 6.2.
The drones are equipped with system-on-chip (SoC) semiconductors: Snapdragon 821 with Quad-core up to 2.15GHz [92]. The mean service time for the two types of messages are calculated as followed:

- The length of safety message is exponentially distributed with parameter $\lambda_1' = \frac{2.15\text{GHz}\times 4}{512\times 8\text{bits}} \approx 2.1 \times 10^6$. Thus, $E[B_1] = \frac{1}{\lambda_1'} = 4.763 \times 10^{-7}$ s, $E[B_1^2] = \frac{1}{(\lambda_1')^2} \approx 2.269 \times 10^{-13}$.
- On the other hand, the length of vehicle state information is a constant and equals to 32 bytes. Thus, $E[B_2] = \frac{32\times 8}{2.15\text{GHz}\times 4} = 2.977 \times 10^{-8}$ s, $E[B_2^2] = 0$.

Table 6.2: Main notations and terminologies

Section	Parameter	Meaning	Value	Reference
	$E[W_{i,j}^{D2V}]$	The D2V propagation delays	$\frac{d'_{i,j}}{r_i}$	
	$E[W_{i,j}^{V2D}]$	The V2D propagation delays	$\frac{d_{i,j}}{r_i}$	

Table 6.2 continued from previous page

Section	Parameter	Meaning	Value	Reference
V2D and D2V Propagation Delay	$d_{i,j}$ and $d'_{i,j}$	The distances between the UAV and the vehicle j from which the request is sent during the V2D and D2V communication	$\sqrt{(x_i - x_j)^2 - (y_i - y_j)^2}$	
	$r_{i,j}$	The achievable propagation rate of the communication lnk	$B * \log_2(1 + \text{SNR}_{i,j})$	
Queuing Delay at the Drone's Side	λ_i	The arrival rate of a class i request	$\lambda_1 = 20\%$ of vehicle number $\lambda_2 = 2.5$	[91]
	$E[B_i]$	The mean service time at the drone of a class i request	$E[B_1] = 4.763 \times 10^{-7}$, $E[B_1^2] = 2.269 \times 10^{-13}$ $E[B_2] = 2.977 \times 10^{-8}$, $E[B_2^2] = 0$	[92]
	$E[W_i]$	The mean waiting time in the queue of a class i request		
	$E[R_i]$	The mean residual service time of a class i request		
	$E[S_i]$	The mean sojourn time of a class i request	$E[S_i] = E[B_i] + E[W_i]$	
	$E[L_i]$	The average number of requests of class i waiting in the queue		
	$E[W_{i,j}]$	The total V2D delay of a class i request from a vehicle j		
	$W_{i,j}^{V2D}$	V2D propagation delay		
	$W_{i,j}^{D2V}$	Drone-to-vehicle (D2V) propagation delay		
	$W_{i,j}^{s}$	Queuing delay at the drone side		
	T_i	The maximum waiting time for request class i	0.2 s	[82]
	a and b	Environmental constant depending on rural or urban area	14.39 0.13	[85]
	h_i	The height of UAV i from ground level	[100,150]	[2]
	$\theta_{i,j}$	Elevation angle between UAV i and vehicle j	$\arctan(\frac{h_i}{d_{i,j}^{hor}})$	

Table 6.2 continued from previous page

Section	Parameter	Meaning	Value	Reference
Energy for D2V Communications	$d_{i,j}^{hor}$	The horizontal distance between the UAV$_i$ and the vehicle$_j$	$\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$	
	$d_{i,j}^{euc}$	The euclidean distance between the UAV i and vehicle j	$\sqrt{h_i^2+\left(d_{i,j}^{hor}\right)^2}$	
	$P_{i,j}(LoS)$	The probability of line-of-sight link	$P_{i,j}(LoS)=\frac{1}{1+a\ \exp(-b(\theta_{i,j}-a))}$	
	$P_{i,j}(NLoS)$	The probability of non-line-of-sight link	$1-P_{i,j}(LoS)$	
	$PL_{i,j}(LoS)$	Path loss with LoS link	$20\log\left(\frac{4\pi f_c d_{i,j}^{euc}}{c}\right)+\eta_{LoS}$	
	$PL_{i,j}(NLoS)$	Path loss with NLoS link	$20\log\left(\frac{4\pi f_c d_{i,j}^{euc}}{c}\right)+\eta_{NLoS}$	
	f_c	Transmit frequency for uplink and downlink of the D2V communication	2.4GHz	[2]
	η_{LoS} and η_{NLoS}	Additional losses for LoS and NLoS links	1 dB 20 dB	[85]
	c	The speed of light	3×10^8 m/s	
	$G(i,j)$	The channel gain of the communication link between the UAV i and vehicle j	$G(i,j)=\frac{1}{PL(i,j)}$	
	$\text{SNR}_{i,j}$	The signal-to-noise ratio	$\frac{p_i G_{i,j}}{\sum_{n\in\mathbb{N}_{\text{int}}} p_n G_{n,j}+N_0}$	
	$C_{i,j}$	The achievable data rate in bits per second (bps)	$B\cdot\log_2(1+\text{SNR}_{i,j})$	
	p_i^{trans}	The transmit power of UAV i	280 mW	[2]
	N_0	The noise power	-174 dB/Hz	[87]
	B	The bandwidth for the D2V communication	100 MHz	[2]
	S_i	The size of the message that UAV i is going to send	512 bytes	[2]
Energy for D2D Communi-	$PL_{n,m}^{D2D}$	The D2D LoS path loss between UAV n and UAV m	$20\log\left(\frac{4\pi f_c d_{n,m}^{euc}}{c}\right)+\eta_{LoS}^{D2D}$	
	η_{LoS}^{D2D}	The mean additional loss of the D2D communication link		
	$d_{n,m}^{euc}$	The euclidean distance between UAV n and UAV m	$\sqrt{(x_n-x_m)^2+(y_n-y_m)^2}$	
	$G_{n,m}^{D2D}$	The channel gain	$\frac{1}{PL_{n,m}^{D2D}}$	

Table 6.2 continued from previous page

Section	Parameter	Meaning	Value	Reference
cations	$SNR_{n,m}^{D2D}$	The signal-to-noise ratio	$\frac{p_n G_{n,m}^{D2D}}{\sum_{i \in \mathbb{N}_{\text{int}}} p_i G_{i,j}^{D2D} + N_0}$	
	$C_{n,m}^{D2D}$	Achievable data rate	$B \cdot \log_2\left(1 + \text{SNR}_{n,m}^{D2D}\right)$	
	$E_{n,m}^{D2D}$	Energy consumption for transmitting a message of size S_n	$\frac{S_n}{C_{n,m}^{D2D}}\ p_n^{trans}$	
	S_n	The transmitted message size	512 bytes	[2]
Energy for Mobility	V_i	Horizontal speed of UAV i Vertical speed of UAV i	30 km/h 10 km/h	
	P_0	Power constant representing the blade profile	84.14 N	[89]
	P_1	Power constant representing the induced power levels in hovering status	88.63 N	[89]
	U_{tip}	The tip speed of the rotor blade	120 m/s	[89]
	v_0	The mean rotor induced velocity in hover	4.03	[89]
	d_0	The fuselage drag ratio	0.6	[89]
	s	Rotor solidity	0.05	[89]
	ρ	Air density	1.225 kg/m^3	[89]
	A	Rotor disc area	0.503 m^2	[89]

6.6 Proposed Drone Optimal Trajectory Prediction (DOT-P) framework

We proposed an algorithm based on ORL. Thanks to the offline training process, we alleviate the drones from hovering for a long time during training which leads to a huge energy consumption. Fig. 6.12 illustrates the working modules of proposed algorithm, which consists of the offline training model and the online prediction module. In order to obtain an efficient and accurate training, it is necessary to perform data preprocessing to the collected dataset. The processed data will then be fed into the RL model for training.

6.6.1 Data preprocessing

The observation dataset undergoes a rigorous data preprocessing and cleansing process before analysis and modelling. This process ensures that the data is in a suitable state for the task.

1. The first step is to check the missing values to identify abnormal columns that need more attention due to missing data.
2. The second step is to check for infinite values. This allows to identify columns with problematic infinite values.

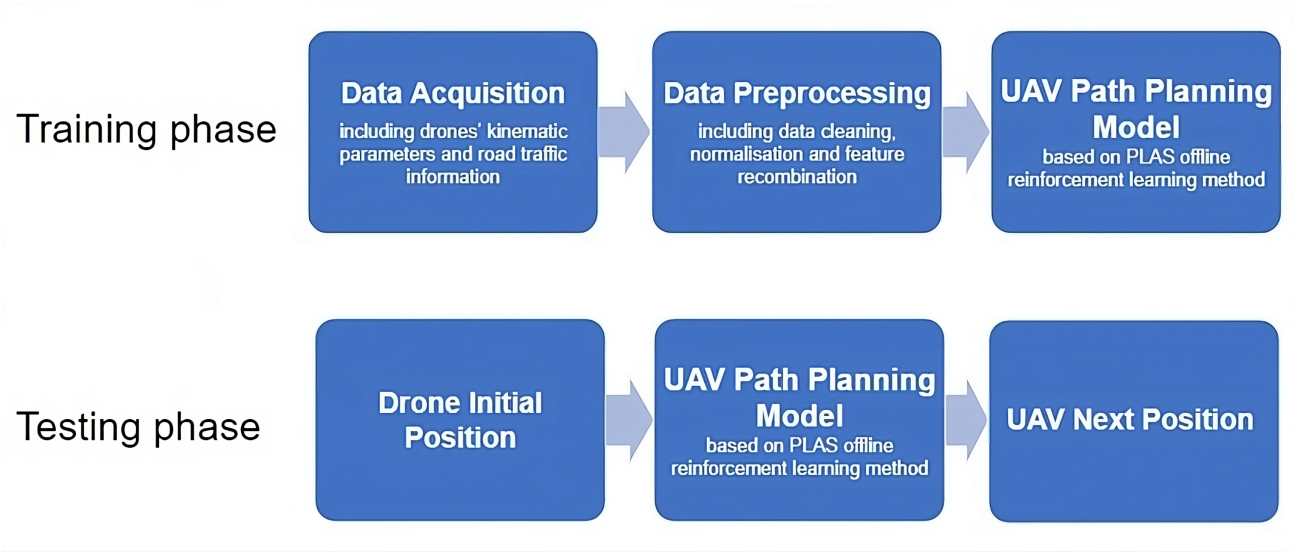

Figure 6.12: Proposed drone optimal trajectory prediction (DOT-P) framework modules

3. After these initial checks, the code proceeds with data imputation. It fills in the missing values with the mean value of their respective columns. This provides a reasonable estimate for missing data and avoids bias in the analysis. It also replaces the infinite values with the maximum non-infinite value in each column. This avoids potential errors or outliers caused by infinite values while retaining valuable data, as illustrated in Fig. 6.13.

4. Finally, it drops the z-axis and unrelated colums for all drones of the observation dataset. This comprehensive data preprocessing and cleansing process improves data quality and prepares the dataset for further analysis, ultimately leading to more accurate and reliable results in data-driven tasks.

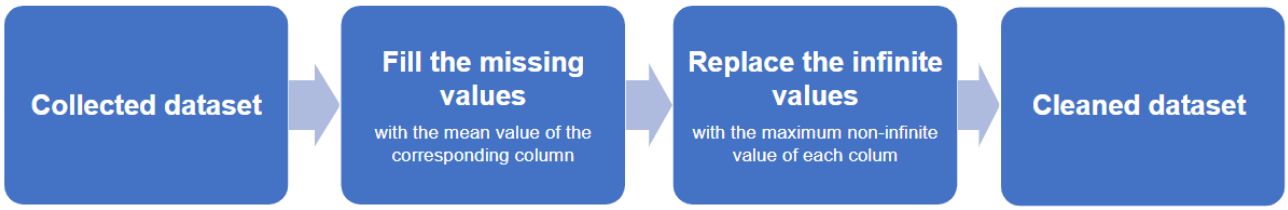

Figure 6.13: Data cleaning process

6.6.2 Input feature

As detailed in section 6.5.4, the collected dataset consists of drones' kinematic parameters and road traffic information, as represented in (6.12) and (6.13). Thus, we define a function that takes the observations dataset and a list of movement column names as inputs. The function' s purpose aims to compute and return the differences between consecutive values in these columns, which represent the displacement between successive data points. The function works as follows: it computes the differences for each pair of coordinates (e.g., dx_1, dy_1, dx_2, dy_2, etc.) and adds them as new columns to the action dataset. Then it sets the index of the action dataset to

match the index of the observation dataset. This ensures that the resulting action dataset is aligned with the original data. This methodology provides a convenient way to calculate and extract actions or changes in coordinates from the observation dataset, which is crucial for our application to track the movements and analyse the changes in spatial data over time.

6.6.3 Reward function

We define the function for drones based on specific input states. The function takes the observation dataset as input and an array of weight coefficients that control the reward calculation. The function performs various computations on the observation dataset to derive the rewards. It first calculates the sum of the connected vehicles for each drone, ρ. Then it computes the mean values of the vehicles' waiting time, W, the drone energy consumption, E, the risky time, t_r, and the blocking time, t_b, for all drones. Moreover, it extracts r, the number of collisions from the observation dataset. The goal is to maximize the vehicular density while minimizing the other parameters (i.e. W, E, t_r, t_b). The total reward R is then computed as in (6.14):

$$R = w_\rho R_\rho - (w_W R_W + w_E R_E + w_{t_r} R_{t_r} + w_{t_b} R_{t_b} + w_r R_r), \tag{6.14}$$

where $R_\rho = \rho$, $R_W = W$, $R_E = E$, $R_{t_r} = t_r$, $R_{t_b} = t_b$, $R_r = r$. And w_ρ, w_W, w_E, w_{t_r}, w_{t_b}, w_r are the weights of each reward, which can be specified and adjusted to customize the reward function. In our work, we set $w_\rho = 3$, $w_W = w_E = w_{t_r} = w_{t_b} = 1.5$, and $w_r = 5$. These values are selected from various configurations. Given that road safety is the paramount concern, the highest weight is assigned to the collision reward, R_r. Regarding the weight of the density reward, w_ρ, as this study focuses on three traffic densities (i.e. sparse, medium and dense), we allocate the second-highest value to w_ρ to underscore the difference. Subsequently, the remaining weights are assigned equal values. It should be noted that in the data preprocessing module, the input data are normalised, leading to normalised reward in (6.14). The reward values are further scaled using the min-max scaler, ranging from 0 to 1.

6.6.4 Agent model based on PLAS (Policy in the Latent Action Space)

To build the drone trajectory prediction model based on Policy in the Latent Action Space (PLAS), we adopt the d3rlpy module, which is a Python library for deep RL that supports various offline algorithms, both discrete and continuous [134]. We compare the various Q-value function supported in d3rlpy in Table. 6.3 in order to choose the most suitable one.

To asses the performance of our proposed algorithm, we evaluate the model with a testing set with the same settings as the training scenario. The testing set consists of the trajectories of each vehicle on the road, their indexes and priorities, as well as the trajectories of each of the four drones. After data preprocessing, the vehicle and drone trajectories will be used to compute the drone energy consumption and vehicle-to-drone E2E delay using predicted trajectory. And one can compare the new values with the original values from the testing set. The testing workflow is shown in Fig. 6.14.

After conducting numerous trials on the training dataset, we identified the optimal configura-

Table 6.3: Supported Q-value algorithms in d3rlpy.

Algorithm	**Support Discrete Control**	**Support Continous Control**	**Support Offline Reinforcement Learning**
Behavior Cloning (supervised learning)	✓	✓	
Neural Fitted Q Iteration (NFQ)	✓	x	✓
Deep Q-Network (DQN)	✓	x	
Double DQN	✓	x	
Deep Deterministic Policy Gradients (DDPG)	x	✓	✓
Twin Delayed Deep Deterministic Policy Gradients (TD3)	x	✓	✓
Soft Actor-Critic (SAC)	✓	✓	✓
Batch Constrained Q-learning (BCQ)	✓	✓	✓
Bootstrapping Error Accumulation Reduction (BEAR)	x	✓	✓
Conservative Q-Learning (CQL)	✓	✓	✓
Advantage Weighted Actor-Critic (AWAC)	x	✓	✓
Critic Reguralized Regression (CRR)	x	✓	✓
Policy in Latent Action Space (PLAS)	x	✓	✓
TD3+BC	x	✓	✓
Implicit Q-Learning (IQL)	x	✓	✓
Decision Transformer	Under development	✓	✓

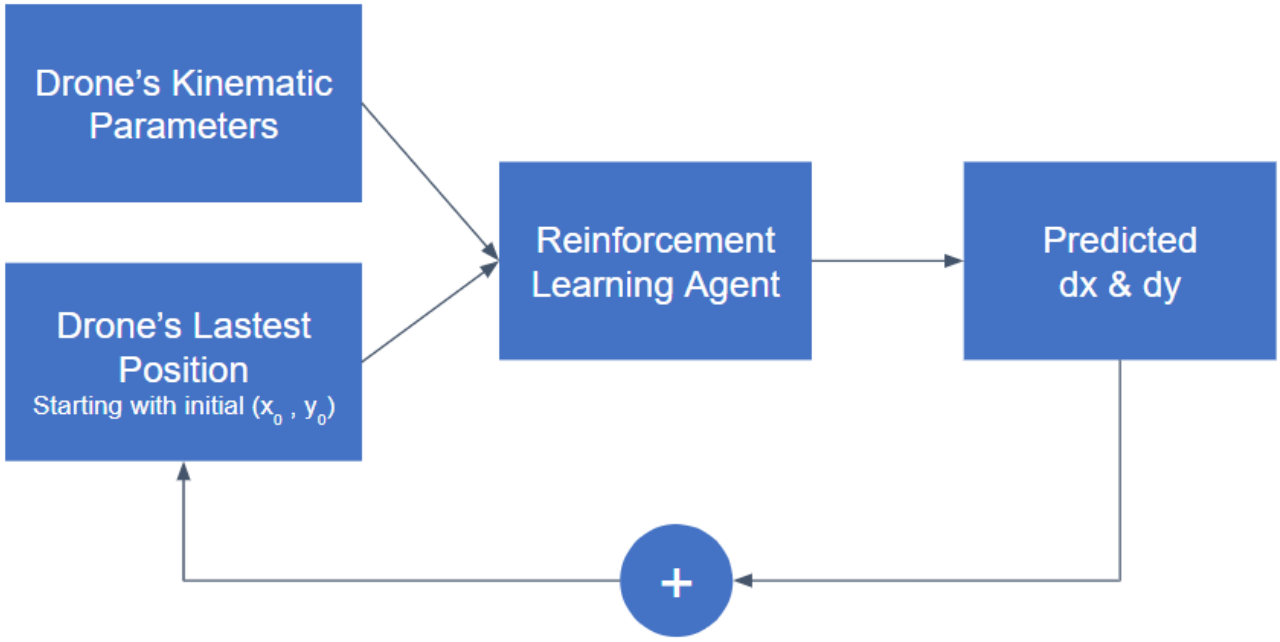

Figure 6.14: Testing workflow for drone position prediction

tions for PLAS with Perturbation, which consists of 30 epochs with 5000 steps per epoch, and for PLAS, which entails 60 epochs with 5000 steps per epoch. The resulting predicted trajectory are visualized in Fig. 6.15 and Fig. 6.16. It should be noted that PLAS stands for Policy in the Latent Action Space. It is a method that learns a policy in a low-dimensional latent space that is mapped to the original action space by a decoder network. PLAS with Perturbation would introduce a small amount of noise or randomness to the latent action space, so that the policy can explore slightly different actions than those in the dataset.

The predicted movement of each drone dx and dy are shown in Fig. 6.17 and Fig. 6.18 it is almost the same between -4 meters and +4 meters.

We employ an unsupervised machine learning algorithm, specifically the K-means algorithm, to determine the predominant areas where each drone self locates the majority of its time. This analysis results in the partitioning of the predicted trajectory into three distinct zones, as illustrated in Fig. 6.19 and Fig. 6.20.

6.6.5 Performance evaluation

We focus on three performance metrics: drone path length, drone energy consumption and V2D E2E delay. To compute these values, we first generate and collect a testing set with the trajectories of all the vehicles on the road, as well as the trajectories of the four drones. Then, the V2D E2E delay and drone energy consumption can be calculated by (6.3) and (6.4). Algorithm. 7 shows the pseudo-code of the parameter computation. The original values and the values using PLAS model with perturbation, and PLAS model are shown in Table. 6.4, 6.5 and 6.6. One can see that PLAS with perturbation yields superior results in terms of path length, average delay, and energy consumption. Moreover, the performance improvement of the three performance metrics averaged from the four drones using PLAS with perturbation compared

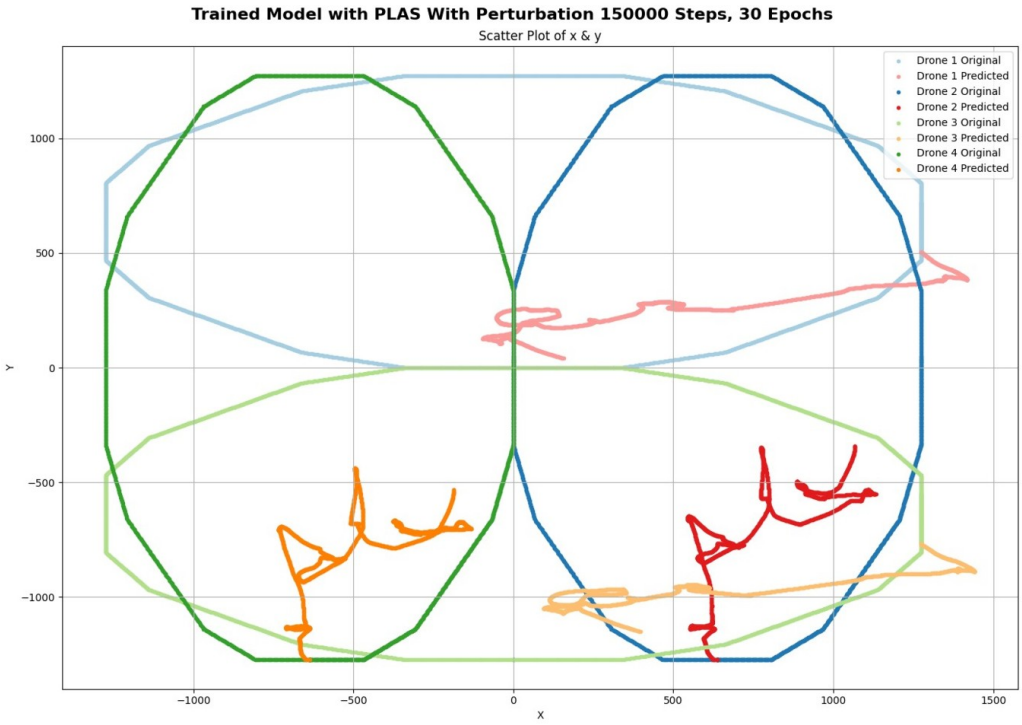

Figure 6.15: The original and the predicted trajectories using PLAS with perturbation

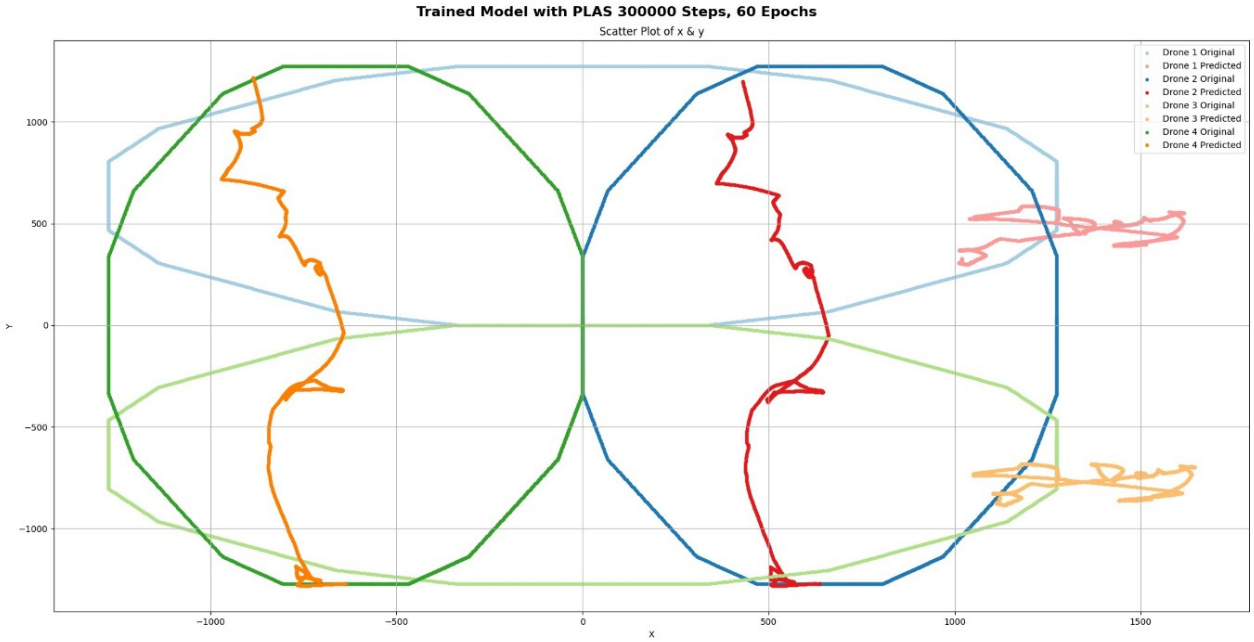

Figure 6.16: The original and the predicted trajectories using PLAS

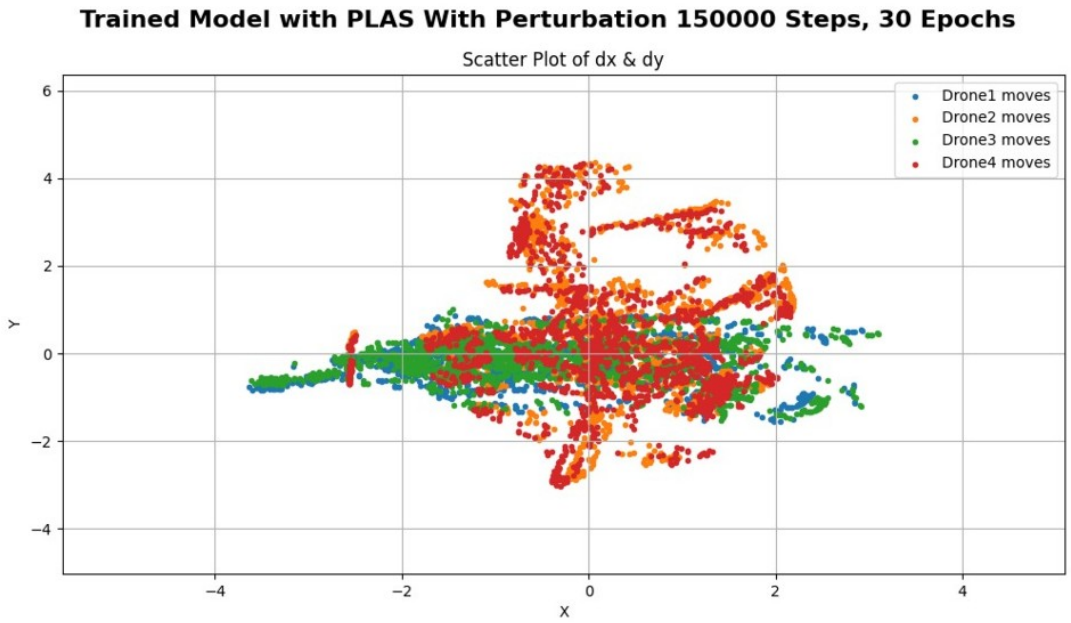

Figure 6.17: The movement of each drone dx and dy using PLAS with perturbation

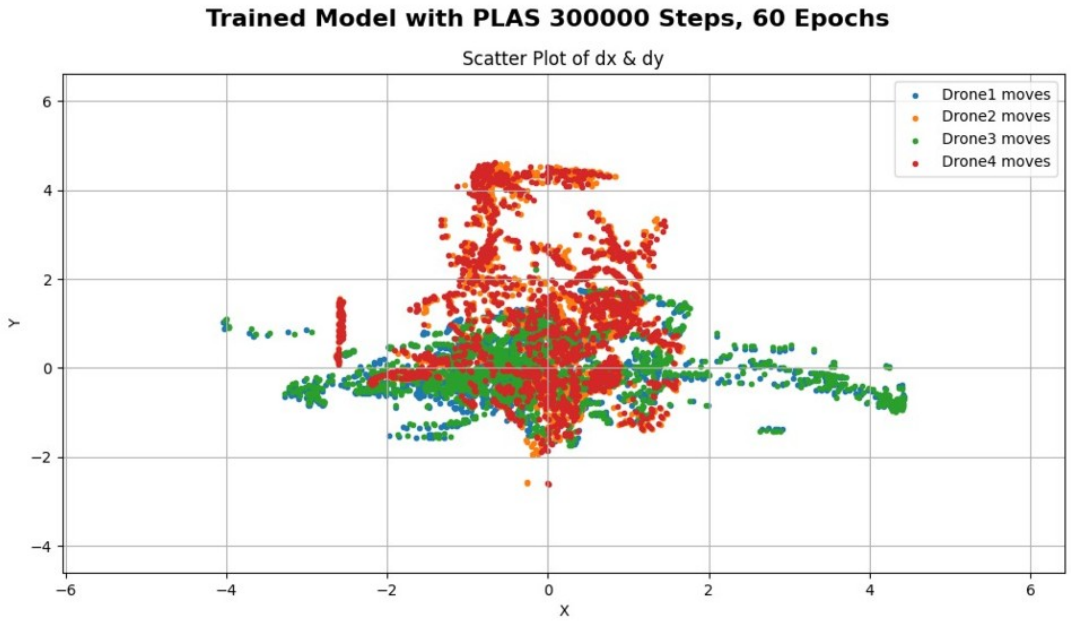

Figure 6.18: The movement of each drone dx and dy using PLAS

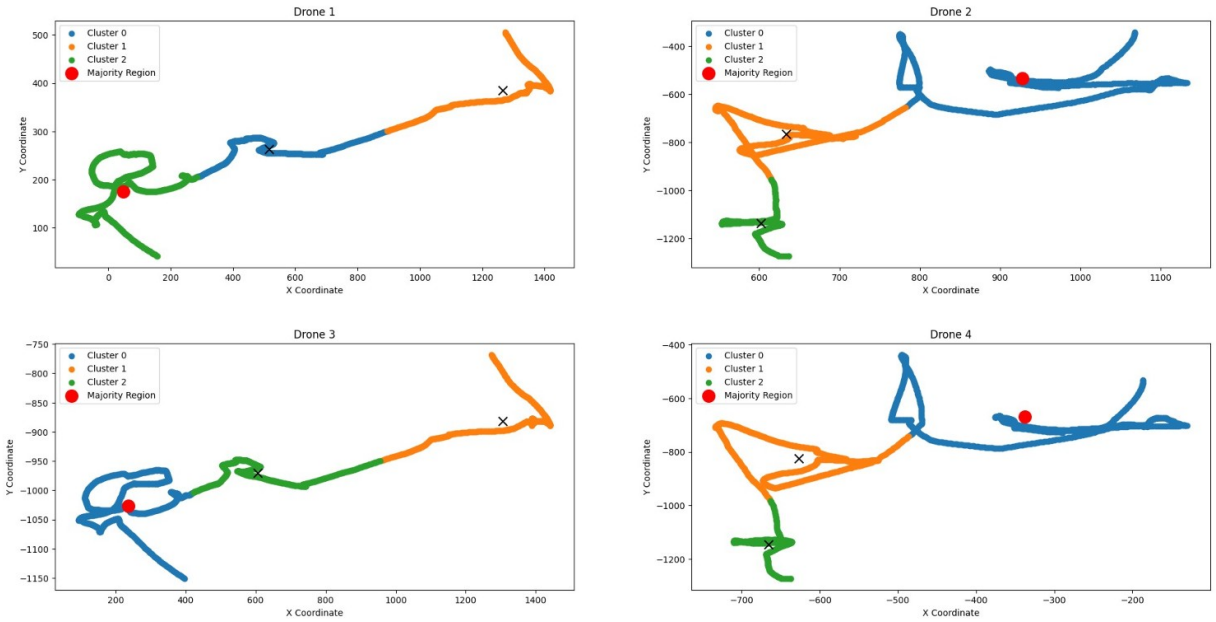

Figure 6.19: K-means clustering applied on the predicted trajectories using the PLAS with perturbation

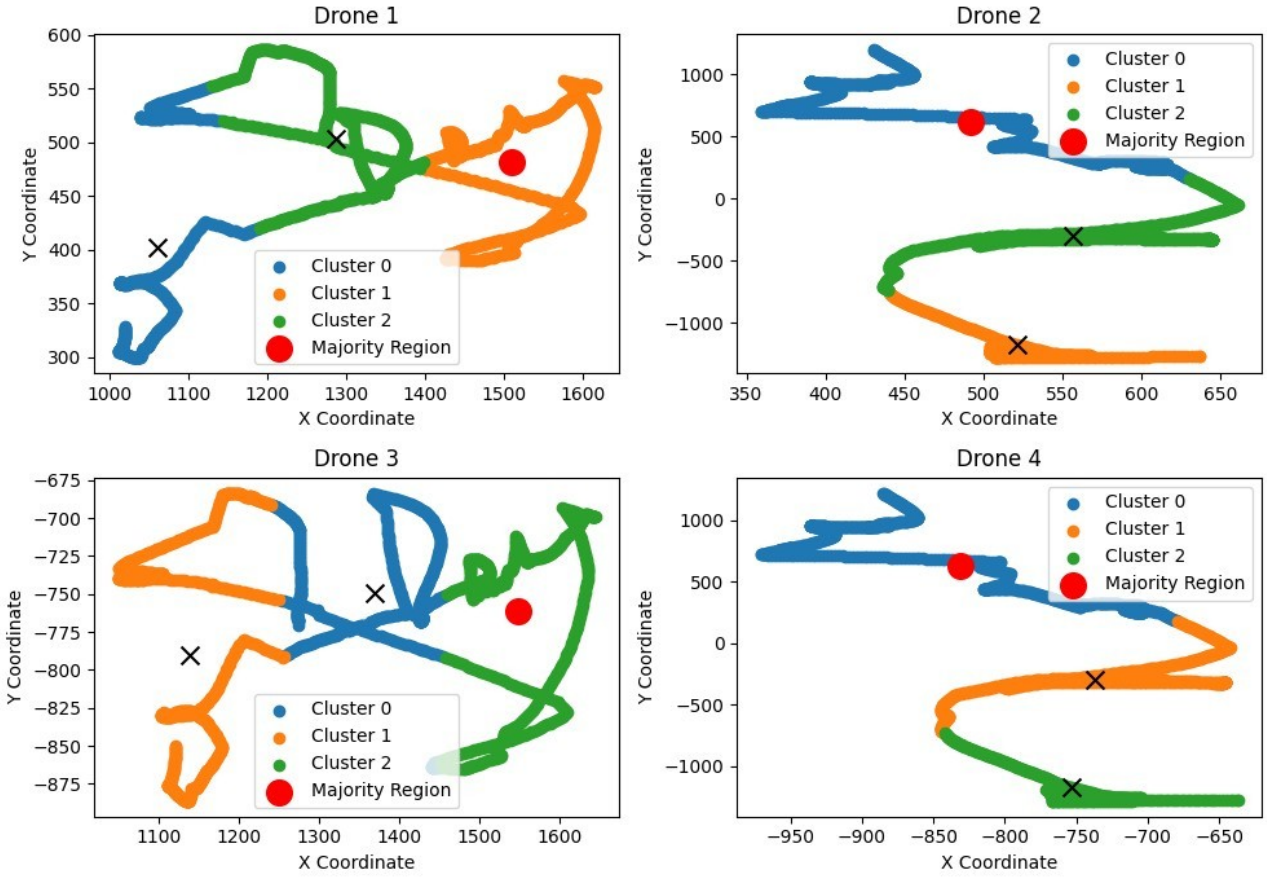

Figure 6.20: K-means clustering applied on the predicted trajectories using the PLAS

Algorithm 7 `Delay and energy consumption evaluation`

Input: Vehicle trajectories and drone trajectories
Output: Average E2E delay and energy consumption for each drone
for each simulation step **do**
    for each drone i, i ∈ [1, 4] **do**
        Update the list of vehicles in its communication range, l_v[i]
        for each vehicle j ∈ l_v[i] **do**
            Compute delay d with function total_delay()
            delay[i] += d
            Compute energy consumption e with function total_energy()
            energy[i] += e
        end for
        tot_delay[i].append(delay[i]/len(l_v[i]))
        tot_energy[i].append(energy[i]/len(l_v[i]))
    end for
end for

with the original values is 58.23%, 49.22% , and 19.7%. And the average improvement using PLAS compared with the original values is 54.64%, 46.22% and 0.81%. These results proves the the fact that the proposed model based on PLAS reduces V2D E2E delay and drone energy consumption.

According to Fig. 6.19 and 6.20, each of the four drones occupies one road segment, which enhances the fact that the drone learns to monitor the road under its coverage.

Table 6.4: Path length comparison between the original values and trained models

Drone Index	Original (m)	PLAS with Perturbation (m)	PLAS (m)
1	6547.4	2789.44	2808.4
2	7959.47	3274.82	3795.5
3	6547.4	2782	2757.36
4	7959.47	3273.5	3798.18

Table 6.5: Average drone energy consumption comparison between the original values and trained models

Drone Index	Original (w)	PLAS with Perturbation (w)	PLAS (w)
1	127.962856	68.39249	70.884233
2	133.527065	65.798664	78.134046
3	120.749584	63.341451	64.175147
4	133.193975	60.886852	64.02225

Table 6.6: Average delay comparison between the original values and trained models

Drone Index	Original (ms)	PLAS with Perturbation (ms)	PLAS (ms)
1	35.334	26.69	28.833
2	32.586	20.868	33.557
3	32.085	30.283	32.923
4	28.39	25.258	32.048

6.6.6 Polynomial-based approach

We also explored an alternative approach focused on identifying optimal points that exhibit the highest responsiveness while offering the most significant rewards. To achieve this, we applied the same reward function as described in section 6.6.3 and (6.14). However, it is important to note that this method is specifically designed for random trajectories with a substantial number of points, as the 100,000-point example illustrated in Fig. 6.21. Subsequently, we pinpointed the most responsive points with the highest rewards. In our case, we selected the top 1% of these points for further analysis, as depicted in Fig. 6.22.

We apply polynomial regression with a 3rd-degree polynomial to effectively fit a curve to the input of the best locations of the drones. Subsequently, we derive a polynomial equation using these coefficients. Afterward, we enhance the data representation by creating a more refined set of points through linear interpolation between the minimum and maximum values of the input locations. This process is crucial for optimizing the trajectories for each drone, as visualized in Fig. 6.23

Performance analysis

Finally, we employ this fitted curve to estimate the approximate length of the path represented by the collection of x and y coordinates. As envisioned in Table. 6.7, we notice that the travelling distance has been reduced by more than 300 times, which implies a reduction in energy consumption.

Table 6.7: Comparison of the paths travelled by each drone

	Original path length (m)	Original coverage length (m)	Approximate length of the predicted path (m0
Drone 1	124973.11	753	381.02
Drone 2	124721.48	2499.06	479.88
Drone 3	124448.89	725.9	344.76
Drone 4	125096.67	2144.23	302.02

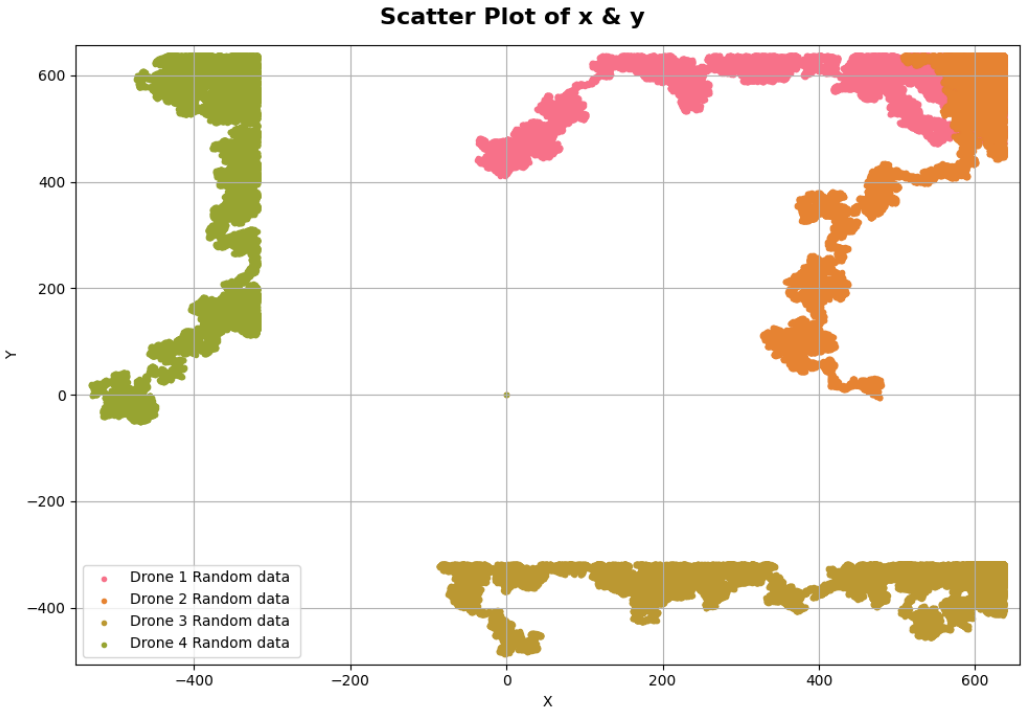

Figure 6.21: Random walk trajectory of the 4 drones

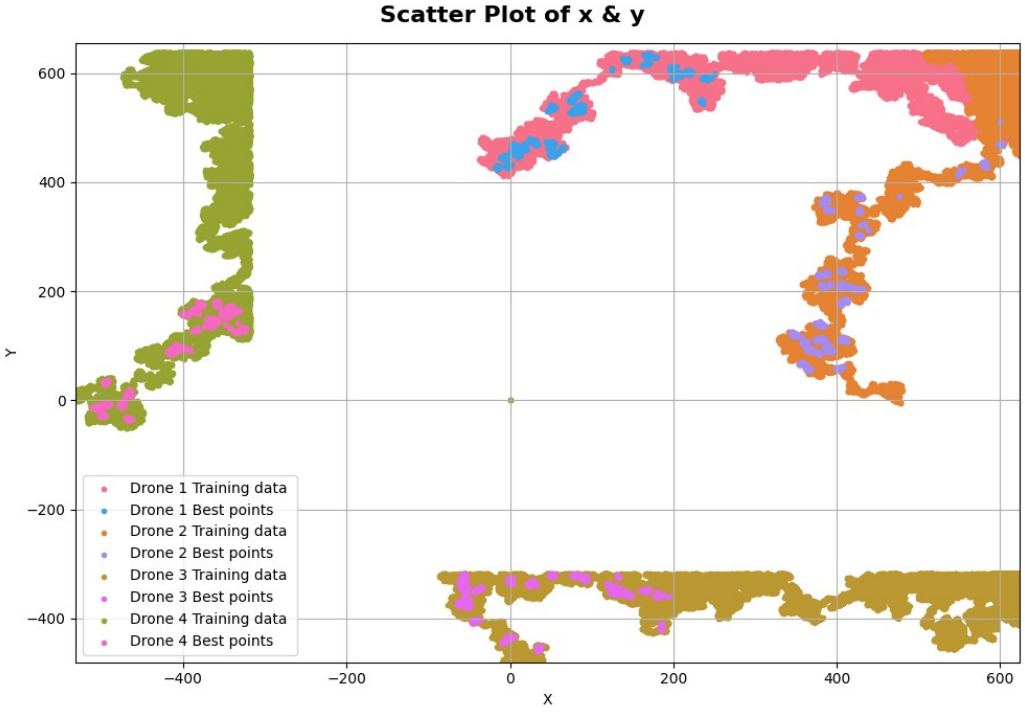

Figure 6.22: Random trajectory of drones and the highest responsive points

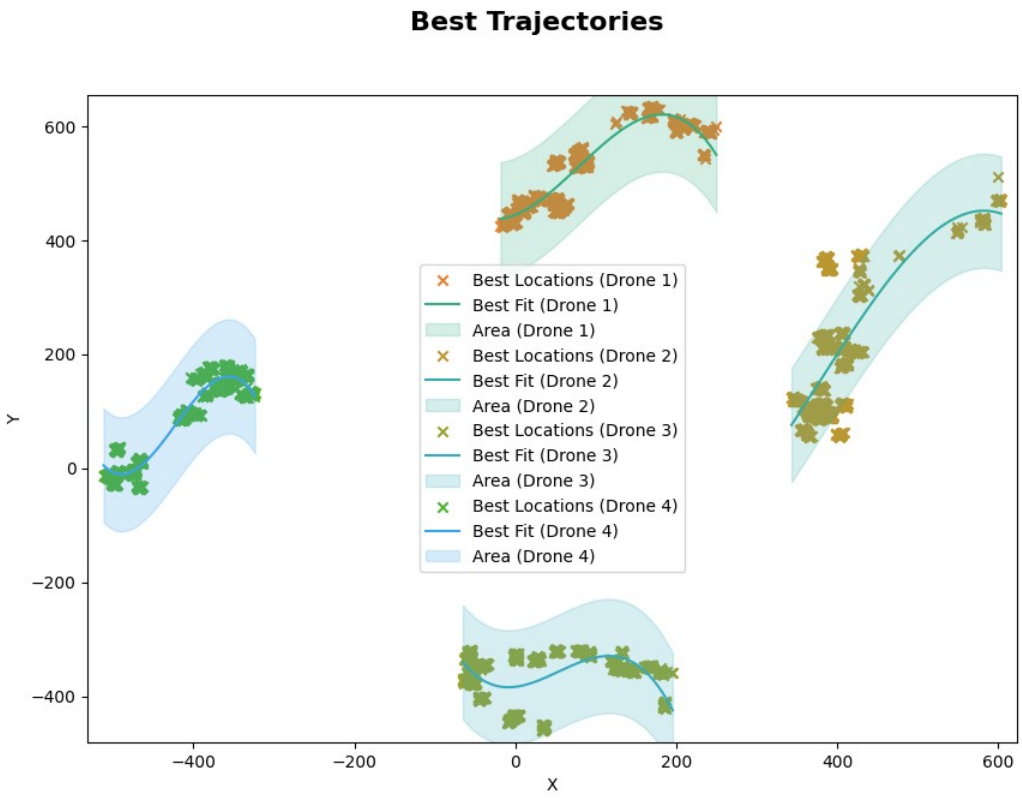

Figure 6.23: Best drone trajectories approximated by a 3rd-degree polynomial

6.7 Conclusion

In this chapter, we present our DOT-P algorithm for optimizing drone trajectories in a dynamic vehicular network. We first conduct an extensive literature study on existing path planning algorithms. Then, we present the details of DOT-P, including the input, output and reward function. Specifically, we base our algorithm on PLAS, an ORL model to prevent time and energy-consuming online training.

Numerical results demonstrate that our method can reduce the travelled path length, the average vehicle communication delay, and the average drone energy consumption by adjusting drones' positions according to the traffic conditions. Moreover, the proposed method provides a novel and efficient solution for enhancing the drone mobility and reliability in complex scenarios.

Furthermore, we extend our efforts to devising a method for calculating the optimal path, and can strategically guide drones through areas with the highest reward points in a random walk scenario. This approach is geared towards identifying the most responsive locations and crafting bespoke trajectories for each drone. In this way, we have successfully contributed to the dominant goal of reducing the overall distance travelled by each drone, ultimately optimizing their operational efficiency.

In the next chapter, we will conclude the thesis work and provide future research directions.

Chapter 7
Conclusion

7.1 Thesis work summary

Lane change, as one of the main reasons of road accident, is an important issue to tackle in vehicular networks. Nevertheless, existing LCA maneuvers based on DRL rely basically on vehicle local information, lacking a global view of the overall traffic. In this context, drones, or UAVs provide a promising extension to the vehicular network services thanks to their high mobility, computing ability and LoS communication links with road vehicles. Consequently, in this thesis, we devote our efforts in devising a safe and efficient LCA maneuver in DAVN.

Chapter 2 is devoted to the introduction of ML basis, including supervised learning, unsupervised learning, RL and FL. In addition, we paid special attention to FRL which combines RL and FL.

Chapter 3 is the literature study of LCA in DAVN. We first present the DAVN basis and shed the light on drone's potential of improving road active safety for vehicular networks.

Then, we review the existing work of LCA maneuver using DRL and other techniques. Specifically, we found that existing work rely solely on local vehicle information (i.e. instantaneous speeds, accelerations, distances) to make lane change decisions, without taking into account the overall traffic state (i.e. the road vehicular density). Moreover, the reward function is not dynamically adapted to the fluctuating traffic conditions.

Besides, after studying the related work about FL and FRL for DAVN, we found that the majority of the papers do not perform the calibration of the global update frequency of the FL algorithm. In addition, the delay processing assessment at the drone level is not well investigated.

Based on these understandings, we found it necessary to devise a LCA maneuver with global information sent by drones. As a result, we propose our GL-DEAR platform. Moreover, we take into consideration the dynamic adjustment of the global update frequency for the FL algorithm, and the non-negligible processing delay at the drone with our proposed DAFL framework.

Chapter 4 introduces the proposed drone assisted LCA platform, GL-DEAR. Firstly, the three

main modules of GL-DEAR is detailed, including *Road with emergency vehicles and risks*; *Data file acquisition and processing* and *Real time lane change decision-making* modules.

Particularly, in the second module, we implement the feature dimension reduction technique, PCA, and compare the performance with the GL-DEAR without using PCA. In fact, our input features consists of vehicle kinematic parameters and road traffic information. In this case, the dimension reduction of input feature will lead to information loss, and hence reduced performance.

Then, the extraction of an authentic lane change model from the authentic NGSIM dataset is explained. With a literature survey, we based the NGSIM lane change model on the XGBoost model. This trained model is applied for ordinary vehicles on the road to recreate the real-world lane change behavior.

Next, we tackle the details of our GL-DEAR platform, which is a DQN with a dynamic reward function that takes into consideration road safety, travel efficiency, and passenger's comfort. The performance is further enhanced by the global control of the drones and the two driving modes possessed by ego vehicle. The simulation scenario includes emergency vehicles and random road risks. Numerical results prove that GL-DEAR successfully achieve collision-less trips on a highway prone to risks and emergency vehicles.

Chapter 5 presents our proposed DAFL drone assisted federated deep reinforcement learning framework. This framework enables the cooperative learning between several ego vehicles of a global lane change model. The learned model helps drivers to achieve safe and real-time lane changes.

In more detail, we first propose a global model aggregation algorithm based on client's reputation for the FL central server. In fact, the client's reputation is computed from its local training reward and the cosine similarity between the local update and the global update.

Then, we perform a detailed mathematical analysis of the E2E delay at the drone consisting of the queuing delay and the D2V propagation delay. The former is formed based on a M/G/1 multi-class preemptive queue. Moreover, we provide an accurate modeling of the total power consumption of the drone, including computation, communication and mobility.

Afterwards, we propose a dynamic adjustment threshold for the FL global update frequency. The thresholds for the update frequency are calibrated according to road safety measurements (i.e., collision rate, risky and impolite driving time on the road) and drone energy consumption. The goal is to achieve the best trade-off between safety and power consumption. Consequently, the drone server can dynamically adapt the global model update frequency according to the road safety and its battery life.

Simulation results prove the efficiency of the proposed DAFL framework.

Chapter 6 presents our DOT-P algorithm for optimizing drone trajectories in a dynamic vehicu-

lar network. The goal is to tackle the drone optimal trajectory planning problem and to enhance the trade-off between drone energy consumption and road safety (i.e. V2D E2E delay).

First, we conduct a comprehensive study of the state-of-the-art of drone trajectory planning techniques. Then, we detail the vehicle E2E delay analysis and drone energy consumption modeling.

Afterwards, a big dataset is pre-collected, including drone's kinematic parameters and road traffic information for the model training. Specifically, we propose two kinds of trajectories for the drone: elliptical trajectory and random walk trajectory.

Then, we adopt the PLAS model for the ORL DOT-P to avoid power-consuming online training. Simulation results show great reduction of the drone energy consumption and vehicle E2E delay using the trained model.

7.2 Future Perspectives

This thesis opens the road to various research axes. Our future perspectives are listed below:

- **Generalisation of the framework to more complex and realistic scenarios:** We study lane change maneuvers in a uniform environment, which allows us to draw several interesting conclusions. In the future, the presence of intersections, pedestrians, and ramps where vehicles coming in and out can be considered. This expansion to more complex scenarios introduces additional challenges such as varying traffic densities, unpredictable pedestrian movements, and the need for efficient navigation through intersections. However, our model's robustness and adaptability enable seamless transferability to these realistic scenarios with the assistance of drones. By incorporating real-time data from drone observations, our model can dynamically adjust lane change strategies to account for changing environmental factors and ensure safe and efficient navigation through complex urban landscapes. Thus, our research not only provides insights into lane change maneuvers in controlled environments but also lays the foundation for the development of intelligent systems capable of operating effectively in dynamic and challenging real-world conditions.
- **Inter-drone handover using game theory:** In [2], it has been established that drones operating within vehicular networks have a notable limitation, with a maximum capacity to connect to 25 vehicles simultaneously. As the number of vehicles within the network exceeds this predetermined threshold, it becomes evident that inter-drone handover will be of paramount importance. This transition of communication responsibility between drones becomes imperative to ensure the uninterrupted flow of data and services, especially in scenarios where vehicular populations are large and dynamic. Looking ahead, it is evident that the issue of inter-drone handover within the domain of DAVN deserves further research and exploration. Understanding and developing efficient strategies for seamless inter-drone handover is essential to get profit from the full potential of drones in enhancing vehicular connectivity. This area should be a focal point for future investigations in the field of DAVN.

- **Federated learning for resource allocation:** Another research axes in vehicular networks is the exploration of vehicle traffic prediction through the adoption of FL. This area of research focuses on the issue of resource allocation within vehicular networks, with a particular emphasis on anticipating traffic patterns and optimizing network resources in accordance with them. With FL's unique ability to leverage local data from various vehicles while preserving privacy, this approach holds the potential to enhance the efficiency of traffic management, improve congestion mitigation strategies, and contribute to a safer and more streamlined vehicular ecosystem [144].

  An example is to apply FRL for resource allocation in vehicular networks during a handover. The goal is to optimize the transmission power of vehicles to ensure seamless connectivity throughout the handover process. By dynamically adjusting transmission power levels, vehicles can maintain stable communication links with roadside units (RSUs) or other vehicles despite transitioning between network coverage areas. FRL offers significant advantages in this context by harnessing the wealth of local data collected by RSUs and vehicles themselves. This decentralized approach allows RSUs to anticipate handover events and allocate resources in real-time within a narrow time window, ensuring uninterrupted connectivity for vehicles as they move through the network. By leveraging FRL, vehicular networks can achieve efficient resource allocation, minimize communication disruptions during handovers, and ultimately enhance the overall reliability and performance of the network.

- **Deploying Graph Convolutional Neural Networks (GCNs) in vehicular networks:** Deploying Graph Convolutional Neural Networks (GCNs) in vehicular network research represents a transformative step towards enhancing the efficiency and safety of modern transportation systems. Vehicular networks are inherently complex, dynamic, and interconnected, making them ideal candidates for GCN applications. By treating vehicles as nodes and their communication interactions as edges in a graph, GCNs can extract valuable insights into traffic patterns, congestion, and even predictive maintenance. This advanced approach allows for real-time analysis of vehicular data, enabling intelligent traffic management, accident prevention, and efficient routing [145]. Furthermore, the deployment of GCNs can contribute to the development of autonomous driving systems and intelligent transportation infrastructure, improving the road safety, sustainability, and ultimately improving the overall driving experience.

Bibliography

[1] USDOT- NHTSA (2009). Traffic Safety Facts- Compilation of Motor Vehicle Crash Data from the Fatality Analysis Reporting System and the General Estimates System.

[2] Shi, Weisen, Haibo Zhou, Junling Li, Wenchao Xu, Ning Zhang, and Xuemin Shen. "Drone assisted vehicular networks: Architecture, challenges and opportunities," IEEE Network, vol. 32, no. 3, pp. 130-137, 2018.

[3] Konečný, Jakub, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik. "Federated optimization: Distributed machine learning for on-device intelligence," arXiv preprint, 2016, arXiv:1610.02527

[4] U.S. Department of Transportation Federal Highway Administration. (2016). Next Generation Simulation (NGSIM) Vehicle Trajectories and Supporting Data. [Dataset]. Provided by ITS DataHub through Data.transportation.gov. Accessed 2022-02-01 from http://doi.org/10.21949/1504477

[5] Wei, X., Yi, Z., Li, W., Zhao, L. and Zhang, W. "Energy harvesting fueling the revival of self-powered unmanned aerial vehicles," Energy Conversion and Management, vol. 283, pp. 116863, 2023.

[6] Anton, Steven R., and Daniel J. Inman. "Vibration energy harvesting for unmanned aerial vehicles." In Active and passive smart structures and integrated systems 2008, vol. 6928, pp. 621-632. SPIE, 2008.

[7] Jiang, Tammy, Jaimie L. Gradus, and Anthony J. Rosellini. "Supervised machine learning: a brief primer." Behavior Therapy, vol 51, no. 5, pp. 675-687, 2020.

[8] Chang, Yin-Wen, Cho-Jui Hsieh, Kai-Wei Chang, Michael Ringgaard, and Chih-Jen Lin. "Training and testing low-degree polynomial data mappings via linear SVM." Journal of Machine Learning Research, vol 11, no. 4, 2010.

[9] Tolles, Juliana, and William J. Meurer. "Logistic regression: relating patient characteristics to outcomes." Jama, vol 316, no. 5, pp. 533-534, 2016

[10] Cortes, C., and V. Vapnik. "Support-vector networks Mach Learn 20." pp. 273-297, 1995.

[11] Sinaga, Kristina P., and Miin-Shen Yang. "Unsupervised K-means clustering algorithm." IEEE access, vol 8, pp. 80716-80727, 2020.

[12] Chandola, Varun, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey." ACM computing surveys (CSUR), vol 41, no. 3, pp. 1-58, 2009.

[13] Jeremy Jordan, "Introduction to autoencoders," https://www.jeremyjordan.me/autoencoders/, Mar 2018.

[14] Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves et al. "Human-level control through deep reinforcement learning." nature, vol 518, no. 7540, pp. 529-533, 2015.

[15] Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves et al. "Human-level control through deep reinforcement learning." nature 518, no. 7540 (2015): 529-533.

[16] Elbir, Ahmet M., Burak Soner, Sinem Çöleri, Deniz Gündüz, and Mehdi Bennis. "Federated learning in vehicular networks." In 2022 IEEE International Mediterranean Conference on Communications and Networking (MeditCom), pp. 72-77. IEEE, 2022.

[17] McMahan, Brendan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. "Communication-efficient learning of deep networks from decentralized data." In Artificial intelligence and statistics, pp. 1273-1282. PMLR, 2017.

[18] Lim, Wei Yang Bryan, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. "Federated learning in mobile edge networks: A comprehensive survey." IEEE Communications Surveys & Tutorials 22, no. 3 (2020): 2031-2063.

[19] Yang, Qiang, Yang Liu, Tianjian Chen, and Yongxin Tong. "Federated machine learning: Concept and applications." ACM Transactions on Intelligent Systems and Technology (TIST) 10, no. 2 (2019): 1-19.

[20] Qi, Jiaju, Qihao Zhou, Lei Lei, and Kan Zheng. "Federated reinforcement learning: Techniques, applications, and open challenges." arXiv preprint arXiv:2108.11887 (2021).

[21] Fan, Xiying, Chuanhe Huang, Bin Fu, Shaojie Wen, and Xi Chen. "UAV-assisted data dissemination in delay-constrained VANETs." Mobile information systems 2018 (2018).

[22] Sedjelmaci, Hichem, Mohamed Ayoub Messous, Sidi Mohammed Senouci, and Imane Horiya Brahmi. "Toward a lightweight and efficient UAV-aided VANET." Transactions on Emerging Telecommunications Technologies 30, no. 8 (2019): e3520.

[23] Raza, Ali, Syed Hashim Raza Bukhari, Farhan Aadil, and Zeshan Iqbal. "An UAV-assisted VANET architecture for intelligent transportation system in smart cities." International Journal of Distributed Sensor Networks 17, no. 7 (2021): 15501477211031750.

[24] C. Sommer, R. German and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," in IEEE Transactions on Mobile Computing, vol. 10, no. 1, pp. 3-15, Jan. 2011, doi: 10.1109/TMC.2010.133.

[25] Lopez, Pablo Alvarez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. "Microscopic traffic simulation using sumo." In 2018 21st international conference on intelligent transportation systems (ITSC), pp. 2575-2582. IEEE, 2018.

[26] N. Zhang, S. Zhang, P. Yang, O. Alhussein, W. Zhuang and X. S. Shen, "Software Defined Space-Air-Ground Integrated Vehicular Networks: Challenges and Solutions," in IEEE Communications Magazine, vol. 55, no. 7, pp. 101-109, July 2017, doi: 10.1109/MCOM.2017.1601156.

[27] Bie, Jing, Mark Roelofsen, Lisheng Jin, and Bart Van Arem. "Lane change and overtaking collisions: Causes and avoidance techniques." Wireless vehicular networks for car collision avoidance (2013): 143-187.

[28] An, HongIl, and Jae-il Jung. "Decision-making system for lane change using deep reinforcement learning in connected and automated driving." Electronics 8, no. 5 (2019): 543.

[29] Mahajan, Vishal, Christos Katrakazas, and Constantinos Antoniou. "Prediction of lane-changing maneuvers with automatic labeling and deep learning." Transportation research record 2674, no. 7 (2020): 336-347.

[30] Shi, Tianyu, Pin Wang, Xuxin Cheng, Ching-Yao Chan, and Ding Huang. "Driving decision and control for automated lane change behavior based on deep reinforcement learning." In 2019 IEEE intelligent transportation systems conference (ITSC), pp. 2895-2900. IEEE, 2019.

[31] Wang, Pin, Ching-Yao Chan, and Arnaud de La Fortelle. "A reinforcement learning based approach for automated lane change maneuvers." In 2018 IEEE Intelligent Vehicles Symposium (IV), pp. 1379-1384. IEEE, 2018.

[32] Mirchevska, Branka, Christian Pek, Moritz Werling, Matthias Althoff, and Joschka Boedecker. "High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning." In 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pp. 2156-2162. IEEE, 2018.

[33] Hoel, Carl-Johan, Krister Wolff, and Leo Laine. "Automated speed and lane change decision making using deep reinforcement learning." In 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pp. 2148-2155. IEEE, 2018.

[34] Wang, Junjie, Qichao Zhang, Dongbin Zhao, and Yaran Chen. "Lane change decision-making through deep reinforcement learning with rule-based constraints." In 2019 International Joint Conference on Neural Networks (IJCNN), pp. 1-6. IEEE, 2019.

[35] Chen, Yilun, Chiyu Dong, Praveen Palanisamy, Priyantha Mudalige, Katharina Muelling, and John M. Dolan. "Attention-based hierarchical deep reinforcement learning for lane change behaviors in autonomous driving." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops, pp. 0-0. 2019.

[36] Wang, Guan, Jianming Hu, Zhiheng Li, and Li Li. "Cooperative lane changing via deep reinforcement learning." arXiv preprint arXiv:1906.08662 (2019).

[37] Y. Liang, Y. Li, A. Khajepour, Y. Huang, Y. Qin, and L. Zheng, "A Novel Combined Decision and Control Scheme for Autonomous Vehicle in Structured Road Based on Adaptive

Model Predictive Control," IEEE T INTELL TRANSP, Vol. 23, no. 9, pp. 16083-16097, September 2022.

[38] Y.S. Son, and W. Kim, "Cooperation-Based Risk Assessment Prediction for Rear-End Collision Avoidance in Autonomous Lane Change Maneuvers," Actuators. Vol. 11, No. 4. MDPI, 2022.

[39] J. Dong, S. Chen, PYJ. Ha, Y. Li, and S. Labi, "A drl-based multiagent cooperative control framework for cav networks: a graphic convolution q network," unpublished.

[40] C. Yang, X. Chen, X. Lin, and M. Li, "Coordinated trajectory planning for lane-changing in the weaving areas of dedicated lanes for Connected And Automated Vehicles," unpublished, 2022.

[41] N. Bao, L. Capito, D. Yang, A. Carballo, C. Miyajima, and K. Takeda, "Data-Driven Risk-Sensitive Control for Personalized Lane Change Maneuvers," IEEE Access, Vol. 10, pp. 36397-36415, April 2022.

[42] J. Wu, W. Huang, N.d. Boer, Y. Mo, X. He, and C. Lv, "Safe Decision-making for Lane-change of Autonomous Vehicles via Human Demonstration-aided Reinforcement Learning," unpublished.

[43] X. He, H. Yang, Z. Hu, and C. Lv, "Robust Lane Change Decision Making for Autonomous Vehicles: An Observation Adversarial Reinforcement Learning Approach," IEEE T-IV, 2022, in press.

[44] S. Samarakoon, M. Bennis, W. Saad and M. Debbah, "Distributed Federated Learning for Ultra-Reliable Low-Latency Vehicular Communications," in IEEE Transactions on Communications, vol. 68, no. 2, pp. 1146-1159, Feb. 2020, doi: 10.1109/TCOMM.2019.2956472.

[45] Kang, Jiawen, Xiong, Zehui, Niyato, Dusit, Zou, Yuze, Zhang, Yang, Guizani, Mohsen. (2019). Reliable Federated Learning for Mobile Networks. IEEE Wireless Communications.

[46] L. WANG, W. WANG and B. LI, "CMFL: Mitigating Communication Overhead for Federated Learning," 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), Dallas, TX, USA, 2019, pp. 954-964, doi: 10.1109/ICDCS.2019.00099.

[47] B. Li, Y. Jiang, W. Sun, W. Niu and P. Wang, "FedVANET: Efficient Federated Learning with Non-IID Data for Vehicular Ad Hoc Networks," 2021 IEEE Global Communications Conference (GLOBECOM), Madrid, Spain, 2021, pp. 1-6, doi: 10.1109/GLOBECOM46510.2021.9685068.

[48] S. Liu, J. Yu, X. Deng and S. Wan, "FedCPF: An Efficient-Communication Federated Learning Approach for Vehicular Edge Computing in 6G Communication Networks," in IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 2, pp. 1616-1629, Feb. 2022, doi: 10.1109/TITS.2021.3099368.

[49] D. Ye, R. Yu, M. Pan and Z. Han, "Federated Learning in Vehicular Edge Computing: A Selective Model Aggregation Approach," in IEEE Access, vol. 8, pp. 23920-23935, 2020, doi: 10.1109/ACCESS.2020.2968399.

[50] J. So, B. Güler and A. S. Avestimehr, "Byzantine-Resilient Secure Federated Learning," in IEEE Journal on Selected Areas in Communications, vol. 39, no. 7, pp. 2168-2181, July 2021, doi: 10.1109/JSAC.2020.3041404.

[51] S. Otoum, I. Al Ridhawi and H. T. Mouftah, "Blockchain-Supported Federated Learning for Trustworthy Vehicular Networks," GLOBECOM 2020 - 2020 IEEE Global Communications Conference, Taipei, Taiwan, 2020, pp. 1-6, doi: 10.1109/GLOBECOM42002.2020.9322159.

[52] H. Yang, J. Zhao, Z. Xiong, K. -Y. Lam, S. Sun and L. Xiao, "Privacy-Preserving Federated Learning for UAV-Enabled Networks: Learning-Based Joint Scheduling and Resource Management," in IEEE Journal on Selected Areas in Communications, vol. 39, no. 10, pp. 3144-3159, Oct. 2021, doi: 10.1109/JSAC.2021.3088655.

[53] Y. Wang, Z. Su, N. Zhang and A. Benslimane, "Learning in the Air: Secure Federated Learning for UAV-Assisted Crowdsensing," in IEEE Transactions on Network Science and Engineering, vol. 8, no. 2, pp. 1055-1069, 1 April-June 2021, doi: 10.1109/TNSE.2020.3014385.

[54] T. Zeng, O. Semiari, M. Mozaffari, M. Chen, W. Saad and M. Bennis, "Federated Learning in the Sky: Joint Power Allocation and Scheduling with UAV Swarms," ICC 2020 - 2020 IEEE International Conference on Communications (ICC), Dublin, Ireland, 2020, pp. 1-6, doi: 10.1109/ICC40277.2020.9148776.

[55] W. Y. B. Lim et al., "Towards Federated Learning in UAV-Enabled Internet of Vehicles: A Multi-Dimensional Contract-Matching Approach," in IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 8, pp. 5140-5154, Aug. 2021, doi: 10.1109/TITS.2021.3056341.

[56] W. Sun, N. Xu, L. Wang, H. Zhang and Y. Zhang, "Dynamic Digital Twin and Federated Learning With Incentives for Air-Ground Networks," in IEEE Transactions on Network Science and Engineering, vol. 9, no. 1, pp. 321-333, 1 Jan.-Feb. 2022, doi: 10.1109/TNSE.2020.3048137.

[57] Z. Cheng, M. Liwang, X. Xia, M. Min, X. Wang and X. Du, "Auction-Promoted Trading for Multiple Federated Learning Services in UAV-Aided Networks," in IEEE Transactions on Vehicular Technology, vol. 71, no. 10, pp. 10960-10974, Oct. 2022, doi: 10.1109/TVT.2022.3184026.

[58] J. S. Ng et al., "Joint Auction-Coalition Formation Framework for Communication-Efficient Federated Learning in UAV-Enabled Internet of Vehicles," in IEEE Transactions on Intelligent Transportation Systems, vol. 22, no. 4, pp. 2326-2344, April 2021, doi: 10.1109/TITS.2020.3041345.

[59] Weiming Zhuang, Xin Gan, Yonggang Wen, and Shuai Zhang. 2023. "Optimizing Performance of Federated Person Re-identification: Benchmarking and Analysis". ACM Trans. Multimedia Comput. Commun. Appl. 19, 1s, Article 38 (February 2023), 18 pages. https://doi.org/10.1145/3531013.

[60] B. Brik, A. Ksentini and M. Bouaziz, "Federated Learning for UAVs-Enabled Wireless Networks: Use Cases, Challenges, and Open Problems," in IEEE Access, vol. 8, pp. 53841-53849, 2020, doi: 10.1109/ACCESS.2020.2981430.

[61] Tianqing, Z., Zhou, W., Ye, D., Cheng, Z. and Li, J., 2021, "Resource allocation in IoT edge computing via concurrent federated reinforcement learning," IEEE Internet of Things Journal, 9(2), pp.1414-1426.

[62] Zhang, M., Wang, S. and Gao, Q., 2020. "A joint optimization scheme of content caching and resource allocation for internet of vehicles in mobile edge computing," Journal of Cloud Computing, 9, pp.1-12.

[63] Xu, Q., Zhang, G. and Wang, J., 2023. Research on Cloud-Edge-End Collaborative Computing Offloading Strategy in the Internet of Vehicles Based on the M-TSA Algorithm. Sensors, 23(10), p.4682.

[64] Huang, J., Wan, J., Lv, B., Ye, Q. and Chen, Y., 2023. "Joint computation offloading and resource allocation for edge-cloud collaboration in internet of vehicles via deep reinforcement learning.," IEEE Systems Journal.

[65] Cui, Y., Du, L., He, P., Wu, D. and Wang, R., 2022, April, "Multi-vehicle intelligent collaborative computing strategy for internet of vehicles," In 2022 IEEE Wireless Communications and Networking Conference (WCNC) (pp. 1647-1652). IEEE.

[66] Shao, S., Su, L., Zhang, Q., Wu, S., Guo, S. and Qi, F., 2023, "Multi task dynamic edge–end computing collaboration for urban Internet of Vehicles," Computer Networks, 227, p.109690.

[67] Liam Hebert, Lukasz Golab, Pascal Poupart, and Robin Cohen. 2023, "FedFormer: Contextual Federation with Attention in Reinforcement Learning," In Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems (AAMAS '23). International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 810–818.

[68] Wang, X., Wang, C., Li, X., Leung, V.C. and Taleb, T., 2020, "Federated deep reinforcement learning for Internet of Things with decentralized cooperative edge caching," IEEE Internet of Things Journal, 7(10), pp.9441-9455.

[69] Seid, A.M., Erbad, A., Abishu, H.N., Albaseer, A., Abdallah, M. and Guizani, M., 2023, "Multi-agent Federated Reinforcement Learning for Resource Allocation in UAV-enabled Internet of Medical Things Networks," IEEE Internet of Things Journal.

[70] J. Hao, R. Naja, D. Zeghlache, "GL-DEAR: Global Dynamic Drone Assisted Lane Change Maneuver for Risk Prevention and Collision Avoidance, " IEEE ICC 2023 - IEEE International Conference on Communications, Rome, Italy, 2023, pp. 6584-6590, doi: 10.1109/ICC45041.2023.10279336.

[71] Jamie Cole, "How Long Do Drone Batteries Last? Increase Battery Life," https://discoveryoftech.com/how-long-do-drone-batteries-last, May 2023.

[72] J. Hao, R. Naja, and D. Zeghlache, "Drone-assisted lane change maneuver using reinforcement learning with dynamic reward function," 2022 18th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), Thessaloniki, Greece, 2022, pp. 314-320, doi: 10.1109/WiMob55322.2022.9941534.

[73] Maćkiewicz, Andrzej, and Waldemar Ratajczak. "Principal components analysis (PCA)." Computers & Geosciences 19, no. 3 (1993): 303-342.

[74] Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel et al. "Scikit-learn: Machine learning in Python." the Journal of machine Learning research 12 (2011): 2825-2830.

[75] Zhenni, L. I., Xinghui HUANG, W. A. N. G. Jiao, and M. U. Tong. "Lane change behavior research based on NGSIM vehicle trajectory data." In 2020 Chinese Control And Decision Conference (CCDC), pp. 1865-1870. IEEE, 2020.

[76] X. Gu, J. Yu, Y. Han, M. Han, and L. Wei, "Vehicle lane change decision model based on Random Forest," ICPICS, pp. 115-120, 2019.

[77] Martínez-Vera, Erik, Pedro Bañuelos-Sánchez, and Gibran Etcheverry. "Lane changing model from NGSIM dataset." In Mexican Conference on Pattern Recognition, pp. 25-34. Cham: Springer International Publishing, 2022.

[78] Ye, Fei, Xuxin Cheng, Pin Wang, Ching-Yao Chan, and Jiucai Zhang. "Automated lane change strategy using proximal policy optimization-based deep reinforcement learning." In 2020 IEEE Intelligent Vehicles Symposium (IV), pp. 1746-1752. IEEE, 2020.

[79] Towers, Mark, Terry, Jordan K, Kwiatkowski, Ariel, Balis, John U., Cola, Gianluca, Deleu, Tristan, Goulão, Manuel, Kallinteris, Andreas, KG, Arjun, Krimmel, Markus, Perez-Vicente, Rodrigo, Pierré, Andrea, Schulhoff, Sander, Tai, Jun Jet, Tan, Andrew Jin Shen, and Younis, Omar G. (2023). Gymnasium (v0.29.1). Zenodo. https://doi.org/10.5281/zenodo.8269265

[80] S. Krauss, "Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics," 1998.

[81] S. Fowler, C. H. Häll, D. Yuan, G. Baravdish and A. Mellouk, "Analysis of vehicular wireless channel communication via queueing theory model," 2014 IEEE International Conference on Communications (ICC), 2014, pp. 1736-1741, doi: 10.1109/ICC.2014.6883573.

[82] Jia, Zehan, et al. "Learning-based queuing delay-aware task offloading in collaborative vehicular networks." ICC 2021-IEEE International Conference on Communications. IEEE, 2021.

[83] A. Al-Hourani, S. Kandeepan and S. Lardner, "Optimal LAP Altitude for Maximum Coverage," in IEEE Wireless Communications Letters, vol. 3, no. 6, pp. 569-572, Dec. 2014, doi: 10.1109/LWC.2014.2342736.

[84] C. -C. Lai, C. -T. Chen and L. -C. Wang, "On-Demand Density-Aware UAV Base Station 3D Placement for Arbitrarily Distributed Users With Guaranteed Data Rates," in

IEEE Wireless Communications Letters, vol. 8, no. 3, pp. 913-916, June 2019, doi: 10.1109/LWC.2019.2899599.

[85] S. Mokhtari, N. Nouri, J. Abouei, A. Avokh and K. N. Plataniotis, "Relaying Data With Joint Optimization of Energy and Delay in Cluster-Based UAV-Assisted VANETs," in IEEE Internet of Things Journal, vol. 9, no. 23, pp. 24541-24559, 1 Dec.1, 2022, doi: 10.1109/JIOT.2022.3188563.

[86] Tran, Dinh-Hieu, Symeon Chatzinotas, and Björn Ottersten. "Throughput Maximization for Backscatter-and Cache-Assisted Wireless Powered UAV Technology." IEEE Transactions on Vehicular Technology 71.5 (2022): 5187-5202.

[87] Zhou, Conghao, et al. "Delay-aware IoT task scheduling in space-air-ground integrated network." 2019 IEEE Global Communications Conference (GLOBECOM). IEEE, 2019.

[88] Wu, Gaoxiang, et al. "Adaptive Edge Caching in UAV-assisted 5G Network." 2021 IEEE Global Communications Conference (GLOBECOM). IEEE, 2021.

[89] Zeng, Yong, Jie Xu, and Rui Zhang. "Energy minimization for wireless communication with rotary-wing UAV." IEEE transactions on wireless communications 18, no. 4 (2019): 2329-2345.

[90] H. Ghazzai, A. Khattab and Y. Massoud, "Mobility and Energy Aware Data Routing for UAV-Assisted VANETs," 2019 IEEE International Conference on Vehicular Electronics and Safety (ICVES), 2019, pp. 1-6, doi: 10.1109/ICVES.2019.8906323.

[91] Oubbati, Omar Sami, et al. "Intelligent UAV-assisted routing protocol for urban VANETs." Computer communications 107 (2017): 93-111.

[92] MODAL AI, "Snapdragon Flight - Best drone support for ROS and PX4," https://www.modalai.com/pages/snapdragon-flight, 2023

[93] A. Wegener et al., "TraCI: an interface for coupling road traffic and network simulators," CNS'08, 2008.

[94] An H, Jung J-i, "Decision-Making System for Lane Change Using Deep Reinforcement Learning in Connected and Automated Driving," Electronics. 2019; 8(5):543. https://doi.org/10.3390/electronics8050543

[95] DJI, "Batterie de Vol Intelligente TB65," https://store.dji.com/fr/product/tb65-intelligent-flight-battery?vid=141051, Jan 2024.

[96] Zhou, Wenxuan, Sujay Bajracharya, and David Held. "Plas: Latent action space for offline reinforcement learning." In Conference on Robot Learning, pp. 1719-1735. PMLR, 2021.

[97] Yang, Yunhong, Xingzhong Xiong, and Yuehao Yan. 2023. "UAV Formation Trajectory Planning Algorithms: A Review" Drones 7, no. 1: 62. https://doi.org/10.3390/drones7010062

[98] Anis Naema Atiyah Rafai, Noraziah Adzhar, Nor Izzati Jaini, and Bingxiao Ding. 2022. A Review on Path Planning and Obstacle Avoidance Algorithms for Autonomous Mobile Robots. J. Robot. 2022 (2022). https://doi.org/10.1155/2022/2538220

[99] J. Wang, Y. Li, R. Li, H. Chen, and K. Chu, "Trajectory planning for UAV navigation in dynamic environments with matrix alignment Dijkstra," Soft Comput, vol. 26, no. 22, pp. 12599–12610, Nov. 2022, doi: 10.1007/s00500-022-07224-3.

[100] D. Mandloi, R. Arya, and A. K. Verma, "Unmanned aerial vehicle path planning based on A* algorithm and its variants in 3d environment," Int J Syst Assur Eng Manag, vol. 12, no. 5, pp. 990–1000, Oct. 2021, doi: 10.1007/s13198-021-01186-9

[101] M. Zimmermann and C. König, "Integration of a visibility graph based path planning method in the ACT/FHS rotorcraft," CEAS Aeronaut J, vol. 7, no. 3, pp. 391–403, Sep. 2016, doi: 10.1007/s13272- 016-0197-0

[102] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces,"IEEE Trans. Robot. Automat., vol. 12, no. 4, pp. 566–580, Aug. 1996, doi: 10.1109/70.508439

[103] W. Tan, Y. Hu, Y. Zhao, W. Li, X. Zhang, and Y. Li, "Mission Planning for Unmanned Aerial Vehicles Based on Voronoi Diagram-Tabu Genetic Algorithm," Wireless Communications and Mobile Computing, vol. 2021, p. e4154787, Nov. 2021, doi: 10.1155/2021/4154787.

[104] E. Magid, R. Lavrenov, and I. Afanasyev, "Voronoi-based trajectory optimization for UGV path planning," in 2017 International Conference on Mechanical, System and Control Engineering (ICMSC), St.Petersburg, Russia: IEEE, May 2017, pp. 383–387. doi: 10.1109/ICMSC.2017.7959506.

[105] H. M. Jayaweera and S. Hanoun, "A Dynamic Artificial Potential Field (D-APF) UAV Path Planning Technique for Following Ground Moving Targets," IEEE Access, vol. 8, pp. 192760–192776, 2020, doi: 10.1109/ACCESS.2020.3032929.

[106] A. MOKRANE, A. C. BRAHAM, and B. CHERKI, "UAV Path Planning Based on Dynamic Programming Algorithm On Photogrammetric DEMs," in 2020 International Conference on Electrical Engineering (ICEE), Sep. 2020, pp. 1–5. doi: 10.1109/ICEE49691.2020.9249903.

[107] M. Ramezani, H. Habibi, J. luis S. Lopez, and H. Voos, "UAV Path Planning Employing MPCReinforcement Learning Method Considering Collision Avoidance." arXiv, Mar. 07, 2023. doi: 10.48550/arXiv.2302.10669.

[108] M. R. Cohen, K. Abdulrahim, and J. R. Forbes, "Finite-Horizon LQR Control of Quadrotors on SE_2(3)," IEEE Robotics and Automation Letters, vol. 5, no. 4, pp. 5748–5755, Oct. 2020, doi: 10.1109/LRA.2020.3010214.

[109] L. Lifen, S. Ruoxin, L. Shuandao, and W. Jiang, "Path planning for UAVS based on improved artificial potential field method through changing the repulsive potential func-

tion," in 2016 IEEE Chinese Guidance, Navigation and Control Conference (CGNCC), Aug. 2016, pp. 2011–2015. doi: 10.1109/CGNCC.2016.7829099.

[110] Q. Liang, H. Zhou, W. Xiong, and L. Zhou, "Improved artificial potential field method for UAV path planning," in 2022 14th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA), Jan. 2022, pp. 657–660. doi: 10.1109/ICMTMA54903.2022.00136.

[111] X. Yuan, "Research on the Limitations of UAV Path Planning Based on Artificial Potential Field Method," in 2022 9th International Forum on Electrical Engineering and Automation (IFEEA), Nov. 2022, pp. 619–622. doi: 10.1109/IFEEA57288.2022.10037827.

[112] L. Chen, I. Mantegh, T. He, and W. Xie, "Fuzzy Kinodynamic RRT: a Dynamic Path Planning and Obstacle Avoidance Method," in 2020 International Conference on Unmanned Aircraft Systems (ICUAS), Sep. 2020, pp. 188–195. doi: 10.1109/ICUAS48674.2020.9213964

[113] C. Ntakolia, K. S. Platanitis, G. P. Kladis, C. Skliros, and A. D. Zagorianos, "A Genetic Algorithm enhanced with Fuzzy-Logic for multi-objective Unmanned Aircraft Vehicle path planning missions," in 2022 International Conference on Unmanned Aircraft Systems (ICUAS), Jun. 2022, pp. 114–123. doi: 10.1109/ICUAS54217.2022.9836068.

[114] Gu, Weibin, Kimon P. Valavanis, Matthew J. Rutherford, and Alessandro Rizzo. "UAV model-based flight control with artificial neural networks: a survey." Journal of Intelligent Robotic Systems 100 (2020): 1469-1491.

[115] Ben Aissa, Sana, and Asma Ben Letaifa. "UAV communications with machine learning: challenges, applications and open issues." Arabian Journal for Science and Engineering 47, no. 2 (2022): 1559-1579.

[116] Yahia, Hazha Saeed, and Amin Salih Mohammed. "Path planning optimization in unmanned aerial vehicles using meta-heuristic algorithms: A systematic review." Environmental Monitoring and Assessment 195, no. 1 (2023): 30.

[117] I. K. Nikolos, E. S. Zografos, and A. N. Brintaki, "UAV Path Planning Using Evolutionary Algorithms," in Innovations in Intelligent Machines - 1, J. S. Chahl, L. C. Jain, A. Mizutani, and M. Sato-Ilic, Eds., in Studies in Computational Intelligence. Berlin, Heidelberg: Springer, 2007, pp. 77–111. doi: 10.1007/978-3-540-72696-8_4.

[118] Puente-Castro, Alejandro, Daniel Rivero, Alejandro Pazos, and Enrique Fernandez-Blanco. "A review of artificial intelligence applied to path planning in UAV swarms." Neural Computing and Applications (2022): 1-18.

[119] baeldung, "Firefly Algorithm | Baeldung on Computer Science," Jan. 25, 2023. https://www.baeldung.com/cs/firefly-algorithm (accessed Aug. 25, 2023)

[120] baeldung, "The Bat Algorithm | Baeldung on Computer Science," Mar. 12, 2023. https://www.baeldung.com/cs/the-bat-algorithm (accessed Aug. 25, 2023).

[121] baeldung, "Cuckoo Search Algorithm | Baeldung on Computer Science," Feb. 23, 2023. https://www.baeldung.com/cs/cuckoo-search (accessed Aug. 25, 2023).

[122] baeldung, "Artificial Bee Colony | Baeldung on Computer Science," Jan. 10, 2023. https://www.baeldung.com/cs/artificial-bee-colony (accessed Aug. 25, 2023).

[123] S. Bouguezzi, "Whale Optimization Algorithm | Baeldung on Computer Science," Aug. 26, 2021. https://www.baeldung.com/cs/whale-optimization-algorithm (accessed Aug. 25, 2023)

[124] M. Theile, H. Bayerlein, R. Nai, D. Gesbert, and M. Caccamo, "UAV Path Planning using Global and Local Map Information with Deep Reinforcement Learning," in 2021 20th International Conference on Advanced Robotics (ICAR), Dec. 2021, pp. 539–546. doi: 10.1109/ICAR53236.2021.9659413

[125] Bayerlein, Harald, Mirco Theile, Marco Caccamo, and David Gesbert. "Multi-UAV path planning for wireless data harvesting with deep reinforcement learning." IEEE Open Journal of the Communications Society 2 (2021): 1171-1187.

[126] Torabi, Faraz, Garrett Warnell, and Peter Stone. "Behavioral cloning from observation." arXiv preprint arXiv:1805.01954 (2018).

[127] G. Swamy, S. Choudhury, J. A. Bagnell, and Z. S. Wu, "Inverse Reinforcement Learning without Reinforcement Learning." arXiv, Jun. 06, 2023. Accessed: Aug. 25, 2023. [Online]. Available: http://arxiv.org/abs/2303.14623

[128] J. Rückin, L. Jin, and M. Popović, "Adaptive Informative Path Planning Using Deep Reinforcement Learning for UAV-based Active Sensing." arXiv, Mar. 03, 2022. doi: 10.48550/arXiv.2109.13570.

[129] Aviral Kumar and Avi Singh, "Offline Reinforcement Learning: How Conservative Algorithms Can Enable New Applications,"The Berkeley Artificial Intelligence Research Blog. http://bair.berkeley.edu/blog/2020/12/07/offline/, Dec 7, 2020 (accessed Sep. 06, 2023)

[130] Prudencio, Rafael Figueiredo, Marcos ROA Maximo, and Esther Luna Colombini. "A survey on offline reinforcement learning: Taxonomy, review, and open problems." IEEE Transactions on Neural Networks and Learning Systems (2023).

[131] Yu, Zhenhua, Zhijie Si, Xiaobo Li, Dan Wang, and Houbing Song. "A novel hybrid particle swarm optimization algorithm for path planning of UAVs." IEEE Internet of Things Journal 9, no. 22 (2022): 22547-22558.

[132] Yu, Rong, Jiefei Ding, Xumin Huang, Ming-Tuo Zhou, Stein Gjessing, and Yan Zhang. "Optimal resource sharing in 5G-enabled vehicular networks: A matrix game approach." IEEE Transactions on Vehicular Technology 65, no. 10 (2016): 7844-7856.

[133] Wubben, Jamie, Christian Morales, Carlos T. Calafate, Enrique Hernández-Orallo, Juan-Carlos Cano, and Pietro Manzoni. "Improving UAV mission quality and safety through topographic awareness." Drones 6, no. 3 (2022): 74.

[134] Seno, Takuma, and Michita Imai. "d3rlpy: An offline deep reinforcement learning library." The Journal of Machine Learning Research 23, no. 1 (2022): 14205-14224.

[135] Al-Mousa, Amjed, Belal H. Sababha, Nailah Al-Madi, Amro Barghouthi, and Remah Younisse. "UTSim: A framework and simulator for UAV air traffic integration, control, and communication." International Journal of Advanced Robotic Systems 16, no. 5 (2019): 1729881419870937.

[136] Schoaba, Vagner, Felipe Eduardo Gomes Sikansi, and Luiz Castelo Branco. "Digital signature for mobile devices: A new implementation and evaluation." International Journal of Future Generation Communication and Networking 4, no. 2 (2011): 23-36.

[137] Geeksforgeeks, "Random Walk (Implementation in Python)," https://www.geeksforgeeks.org/random-walk-implementation-python/, October 2017.

[138] Wu, Yang, Guyu Hu, Fenglin Jin, and Jiachen Zu. "A satellite handover strategy based on the potential game in LEO satellite networks." IEEE Access 7 (2019): 133641-133652.

[139] Mouawad, N., Khoder, R., Naja, R. et al. Vertical group handover congestion game for a vehicular platoon in VLC networks. Ann. Telecommun. 77, 601–610 (2022). https://doi.org/10.1007/s12243-021-00885-5

[140] A. Awada, B. Wegmann, I. Viering, and A. Klein, " A game-theoretic approach to load balancing in cellular radio networks,"in IEEE International Symposium on Personal, Indoor and Mobile Radio Communications Conference, Istanbul, Turkey, September 2010.

[141] Li, Bowen, and Junting Chen. "Handover Game for Data Transportation over Dynamic UAV Networks with Predictable Channels." In GLOBECOM 2022-2022 IEEE Global Communications Conference, pp. 3724-3729. IEEE, 2022.

[142] Alhabo, Mohanad, Li Zhang, Naveed Nawaz, and Hayder Al-Kashoash. "Game theoretic handover optimisation for dense small cells heterogeneous networks." IET Communications 13, no. 15 (2019): 2395-2402.

[143] J. Hao, R. Naja, D. Zeghlache, "Adaptive Federated Reinforcement Learning for Critical Realtime Communications in UAV Assisted Vehicular Networks," submitted to Computer Networks, September 2023.

[144] Taik, Afaf, Zoubeir Mlika, and Soumaya Cherkaoui. "Clustered vehicular federated learning: Process and optimization." IEEE Transactions on Intelligent Transportation Systems 23, no. 12 (2022): 25371-25383.

[145] Qiu, Han, Qinkai Zheng, Mounira Msahli, Gerard Memmi, Meikang Qiu, and Jialiang Lu. "Topological graph convolutional network-based urban traffic flow and density prediction." IEEE transactions on intelligent transportation systems 22, no. 7 (2020): 4560-4569.

ECOLE
DOCTORALE

Titre : APPRENTISSAGE POUR LA SURETÉ DANS LES RÉSEAUX VÉHICULAIRES

Mots clés : Apprentissage machine, changement de voie, réseaux vehiculaires, apprentissage fédéré, analyse des délais, contrôle

Résumé : Cette thèse porte sur le développement d'une manœuvre d'aide au changement de voie (lane Change Assistance, LCA) sûre et efficace dans le contexte des réseaux de véhicules assistés par drones (Drone Assisted Vehicular Network, DAVN). En effet, les changements de voie contribuent de manière significative aux accidents de la route, nécessitant des solutions efficaces au sein des réseaux routiers. Les LCA stratégies actuelles établies sur l'apprentissage par renforcement profond (Deep Reinforcement Learning, DRL) sont limitées par les informations locales sur les véhicules, négligeant une vue globale, comme des conditions de circulation. Pour résoudre ce problème, les véhicules aériens sans pilote (Unmanned Aerial Vehicles, UAVs), ou drones, présentent une extension prometteuse des services de réseau automobile grâce à leur mobilité, capacités informatiques et liaisons de communication en visibilité directe (Line-if-Sight, LoS) avec les véhicules routiers.

Dans un premier temps, nous faisons une étude bibliographique sur LCA au sein du DAVN, mettant en évidence le potentiel des drones pour améliorer la sécurité routière. Les approches LCA existantes s'appuient principalement sur des informations locales sur les véhicules et ne prennent pas en compte l'état global du trafic. Afin de réduire cette limitation, nous proposons le GL-DEAR : joint global and local drone-assisted lane change platform based on Deep-Q Network (DQN) with a dynamic reward function, for LCA with drones' assistance.

La plateforme proposée se compose de trois modules : route à risques aléatoires et véhicules d'urgence ; acquisition et traitement des données; prise de décision de changement de voie en temps réel. La manœuvre de changement de voie est basée sur un Deep Q-Network avec des fonctions de récompense dynamiques. Plus précisément, nous adoptons les modèles de changement de voie authentiques basés sur l'ensemble de données NGSIM pour les véhicules routiers ordinaires afin de recréer les comportements de changement de voie du monde réel dans les simulations. Les résultats numériques démontrent la capacité de la plateforme à réaliser des trajets sans collision sur des autoroutes à risque avec des véhicules d'urgence.

Dans un deuxième temps, nous identifions un manque de calibrage de la fréquence de mise à jour globale des algorithmes d'apprentissage fédéré (Federated Learning, FL) et l'absence d'évaluation approfondie du délai de traitement au niveau du drone. Nous proposons donc un cadre d'apprentissage par renforcement fédéré (FRL) assisté par drone, DAFL. Ce cadre permet un apprentissage coopératif entre les véhicules de l'ego en appliquant FL. Il comprend un algorithme d'agrégation de modèles global basé sur la réputation du client et une analyse complète du délai de bout en bout (End-to-End, E2E) au niveau du drone. Plus précisément, la fréquence globale de mise à jour est ajustée dynamiquement en fonction des mesures de sécurité routière et de la consommation énergétique des drones, ce qui donne des résultats efficaces dans les simulations.

Dans la troisième étape, nous concevons l'algorithme DOP-T pour optimiser les trajectoires des drones dans les réseaux de véhicules dynamiques. Cet algorithme vise à équilibrer la consommation énergétique des drones et la sécurité routière. Nous fournissons un état de l'art complet des techniques existantes de planification de trajectoire de drones. Ensuite, sur la base de la modélisation du délai E2E du véhicule et de la modélisation de la consommation d'énergie du drone. Dans la seconde étape, nous formons un modèle d'apprentissage par renforcement hors ligne (Offline-Reinforcement Learning, ORL) pour éviter une formation en ligne consommatrice d'énergie. Les résultats de la simulation démontrent une réduction significative de la consommation d'énergie des drones et du délai E2E du véhicule à l'aide du modèle entraîné.

Title : MACHINE LEARNING FOR ROAD ACTIVE SAFETY IN VEHICULAR NETWORKS

Keywords : Machine learning, lane change, vehicular networks, federated learning, delay analysis, control

Abstract : This thesis focuses on the development of a safe and efficient LCA maneuver in the context of drone-assisted vehicle networks (DAVN). In fact, lane change maneuvers contribute significantly to road accidents, requiring effective solutions within road networks. Current lane change assistance (LCA) strategies relying solely on deep reinforcement learning (DRL) are limited by local vehicle information, neglecting a global view of traffic conditions. To address this problem, unmanned aerial vehicles (UAVs), or drones, present a promising extension of automotive network services due to their mobility, computing capabilities, and line-of-sight (LoS) communications links with road vehicles.

In the first step, we conduct a literature review on LCA within DAVN, highlighting the potential of drones to enhance road safety. Existing LCA approaches predominantly rely on local vehicle information and fail to consider overall traffic states. To address this limitation, we propose the GL-DEAR: joint global and local drone-assisted lane change platform based on Deep-Q Network (DQN) with a dynamic reward function, for LCA with drones' assistance.

The proposed platform consists of three modules: road with random risks and emergency vehicles; data file acquisition and processing; and real-time lane change decision-making. The lane change maneuver is based on a Deep Q-Network with dynamic reward functions. Specifically, we adopt the authentic NGSIM dataset-based lane change models for ordinary road vehicles to recreate real world lane change behaviors in the simulations. Numerical results demonstrate the platform's ability to achieve collision-free trips on risky highways with emergency vehicles.

In the second step, we identify a lack of calibration for the global update frequency in FL algorithms and the absence of thorough drone-level processing delay assessment. To this end, we propose the drone assisted Federated Reinforcement Learning (FRL)-based LCA framework, DAFL. This framework enables cooperative learning between ego vehicles by applying Federated Learning (FL). It includes a client reputation-based global model aggregation algorithm and a comprehensive analysis of End-to-End (E2E) delay at the drone. Specifically, the global update frequency is dynamically adjusted according to road safety measurements and drone energy consumption, yielding efficient results in simulations.

In the third step, we devise the DOP-T algorithm for optimizing drone trajectories in dynamic vehicular networks. This algorithm aims to balance drone energy consumption and road safety. We provide a comprehensive state-of-the-art review of the existing drone trajectory planning techniques. Then, based on the vehicle E2E delay modeling and the drone energy consumption modeling in the second step, we train a Offline Reinforcement Learning (ORL) model to avoid power-consuming online training. Simulation results demonstrate a significant reduction in drone energy consumption and vehicle E2E delay using the trained model.

Institut Polytechnique de Paris
91120 Palaiseau, France