Sorbonne University

Doctoral School of Informatics, Telecommunications and Electronics of Paris

EURECOM

On the Networking of Knowledge in Vehicular Networks

Presented by Duncan Deveaux

Dissertation for Doctor of Philosophy in Informatics, Telecommunications and Electronics

Directed by Prof. Jérôme Härri

Publicly presented and defended on 8 December 2021

A jury committee composed of:

Prof. André-Luc Beylot	INP/ENSEEIHT – IRIT	Reviewer
Prof. Antonella Molinaro	CentraleSupelec	Reviewer
Dr. Onur Altintas	Toyota Motor North America R&D	Examiner
Dr. Chadi Barakat	Inria Sophia Antipolis	Examiner
Dr. Raphaël Troncy	EURECOM	Examiner
Prof. Jérôme Härri	EURECOM	Examiner

Abstract

Knowledge is an abstraction of information, obtained through *experience*. Through the analysis of large amounts of observations, models of the environment can be formed to convey the key patterns and understandings from the underlying descriptive sets of information. In vehicular networks, *connected vehicles*, which generalized information exchange among vehicles and infrastructure, are evolving to *intelligent vehicles*, which implement smart applications such as highly-automated driving, requiring complex decision making based on the current driving context. For example, upon receiving a warning notification about an obstacle on the road, how should a highly automated vehicle react? This decision making requires personalized knowledge which is adapted to the current context of driving, e.g., the type of intersection in which the obstacle was detected.

As such, *knowledge* is taking a key place in the development of smart applications for intelligent vehicles. In addition to producing information, vehicles require knowledge building which is personalized to their driving context. Yet, currently, each automaker is responsible for their own knowledge building and sharing. Instead, what if vehicles could cooperatively build and share knowledge, rather than recreate it? While the distribution of information in vehicular networks has been extensively studied and even standardized, no mechanisms have been defined to name, store, and network knowledge in a mutually understandable manner, to allow the transition from a network of information to a network of knowledge.

In turn, this doctoral work defines a conceptual framework, named *Vehicular Knowledge Networking*, to support knowledge distribution in vehicular networks, and evaluates it in several case studies. Namely, the three complementary aspects of (i) knowledge description, (ii) knowledge storage, and (iii) knowledge networking are described. Knowledge description defines the semantic requirements to describe the interface and the context of application of knowledge in a mutually-understandable manner. In turn, a knowledge layer is defined which interfaces applications and information storage on-board connected vehicles. Then, request messages are conceptualized which let nodes of the vehicular networks synchronize to perform cooperative and distributed knowledge discovery, knowledge application and knowledge building tasks.

On the one hand, the performance of the framework is evaluated for knowledge application, which evaluates the distribution of fully trained knowledge in the right context. Namely, simulations show that the application of a knowledge model by a remote node reduces the overhead compared to the traditional transmission of a full copy of the model for a local application. What is more, in an evaluation of knowledge distribution in a realistic use case of exit probability estimation for vehicles in a roundabout, we use context-describing semantics to apply the knowledge models whose training context is *similar* to the requested context of application. In

turn, we show a significant increase in the accuracy or precision of the produced exit probability knowledge.

On the other hand, the performance of the framework is evaluated for knowledge building, which evaluates the context-aware orchestration of cooperative knowledge training tasks among connected vehicles. We show significant improvements in the training speed and accuracy of a knowledge model being cooperatively trained by distributed vehicles, by reasoning on the context of training of each vehicle and choosing the vehicles which feature the most relevant driving context to take part in training. Moreover, using the case study of roundabout exit probability knowledge, we show an increased accuracy of models trained for new roundabouts for which limited training data is available, when the training data is completed with data extracted from roundabouts which feature a similar training context.

Acknowledgements

In the experience of this 3-year PhD study, I worked on interesting and inspirational topics. The completion of this dissertation would not have been possible without the help of many people, to whom I would like to extend my sincere thanks.

I would like to thank my supervisor, Prof. Jérôme Härri, for his perspective and advice throughout this 3-year journey, which has been highly valuable to keep moving in the right direction, as well as my coworkers from the automotive team in EURECOM. I would also like to extend my gratitude to my colleagues Dr. Onur Altintas, Dr. Takamasa Higuchi, Dr. Seyhan Uçar, Dr. Chang-Heng Wang from Toyota Motor North America R&D, InfoTech Labs on the other side of the Atlantic, who provided guidance and valuable reviews for the works we co-authored. I would like to further thank Toyota InfoTech Labs for their contribution to the funding of our research projects, and EURECOM for funding my PhD studies. I also wish to thank my committee, and my special thanks go to the reviewers, Prof. André-Luc Beylot and Prof. Antonella Molinaro, for using their time to review this manuscript.

Moreover, I wish to thank my friends and family for their support and interest in my research activities, and for sharing valuable time with me during the sanitary lockdowns which punctuated my PhD studies. Finally, I wish to extend my gratitude to my wife Hanna for her continuous love and support throughout the challenging times of my research studies.

Contents

List of Figures

List of Tables

List of abbreviations

AI	Artificial Intelligence
ANN	Artificial Neural Network
AODV	Ad-hoc On-demand Distance Vector
BSA	Basic Set of Applications
BSM	Basic Safety Message
BSS	Basic Service Set
BTP	Basic Transport Protocol
CAM	Cooperative Awareness Message
CAV	Connected and Autonomous Vehicle
CC	Cloud Computing
CCA	Cooperative Collision Warning
CDN	Content Delivery Network
CPM	Collective Perception Message
CV	Connected Vehicle
DENM	Decentralized Environmental Notification Message
DL	Deep Learning
DML	Distributed Machine Learning
DSR	Dynamic Source Routing
DSRC	Dedicated Short Range Communications
EvoC	Evolutionary Computation
FCN	Function-Centric Networking
FIB	Forwarding Information Base
FL	Federated Learning
FuzzL	Fuzzy Logic

GD	Gradient Descent
GL	Gossip Learning
GNSS	Global Navigation Satellite Systems
HAV	Highly-Automated Vehicle
HCM	Highway Capacity Manual
HyperMAS	Hypermedia Multi-Agent Systems
ICN	Information-Centric Networking
IETF	Internet Engineering Task Force
IoT	Internet of Things
IoV	Internet of Vehicles
ITS	Intelligent Transport Systems
KA	Knowledge Acquisition
KB	Knowledge Base
KBS	Knowledge-Based Systems
KCN	Knowledge-Centric Networking
KG	Knowledge Graphs
KIF	Knowledge Interchange Format
KLR	Knowledge Representation Learning
LDM	Local Dynamic Map
MAC	Medium Access Control
MADM	Multi-Agent Decision Making
MANET	Mobile Ad-Hoc Network
MARL	Multi-Agent Reinforcement Learning
MEC	Mobile Edge Computing
MH	Meta-Heuristics
MI	Mutual Information
ML	Machine Learning
MoST	Monaco SUMO Traffic
MSE	Mean Square Error

NDN	Named Data Networking
NFE	NDN Forwarding Engine
NFN	Named Function Networking
OBU	On-Board Unit
OFDM	Orthogonal Frequency-Division Multiplexing
OSI	Open Systems Interconnection
OWL	Web Ontology Language
PIT	Pending Interests Table
PSM	Personal Safety Message
QoS	Quality of Service
RDF	Resource Description Framework
RL	Reinforcement Learning
RMSE	Root Mean Square Error
RPG	Reference Point Group mobility model
RSD	Relative Standard Deviation
RSU	Road-Side Unit
RTK	Real Time Kinematic
RWP	Random WayPoint mobility model
SI	Swarm Intelligence
SL	Supervised Learning
SUMO	Simulation of Urban MObility
TD	Thing Description
TET	Time-Exposed TTC
TraCI	Traffic Control Interface
TTC	Time To Collision
UC	Uncertainty Coefficient
UL	Unsupervised Learning
V2I	Vehicle To Infrastructure
V2N	Vehicle To Network

V2P	Vehicle To Pedestrian
V2V	Vehicle To Vehicle
V2X	Vehicle To Everything
VANET	Vehicular Ad-Hoc Network
VC	Vehicular Clouds
VEC	Vehicular Edge Computing
VKN	Vehicular Knowledge Networking
VMC	Vehicular Micro-Clouds
VRU	Vulnerable Road User
VSS	Vehicle Signal Specification
VSSo	Vehicle Signal Specification Ontology
W3C	World Wide Web Consortium
WAVE	Wireless Access in Vehicular Environments
WoT	Web of Things
WSMP	Wave Short Message Protocol
ZRP	Zone Routing Protocol

Chapter 1

Introduction

1.1 General Introduction

1.1.1 From Information to Knowledge

Over the last decade, we have witnessed the evolution of vehicular networking from 'Vehicular Ad-Hoc Networks' (VANETs), enabling spontaneous direct communications between vehicles, to 'Connected Vehicles' generalizing information exchange among vehicles and infrastructure. In early applications, connected vehicles were networked with the infrastructure, in order to receive information to assist a human driver, such as (i) navigation information, e.g., maps, road or parking data, (ii) information on topics such as weather or traffic flow, (iii) road-related information, e.g., gas station opening times, or (iv) multimedia contents for user infotainment. Such applications allowed the presentation of dynamic information produced by the infrastructure to drivers of connected vehicles, to assist their navigation. Figure 1.1 illustrates the infrastructure in vehicular networks, which involves distant centralized data centers, and *edge units* providing computing and information storage capabilities on the side of the roads, closer to the connected vehicles.

To support the distribution of information from the infrastructure to connected vehicles, enabling technologies such as 5G and even 6G have been developed and applied to an Internet of vehicles, as surveyed, respectively, in [1] and [2]. They allow to scale the distribution of information to the ever increasing number of connected vehicles on the road. According to a report of the Automotive Edge Computing Consortium, all new vehicles produced in the USA by 2025 will be connected [3].

With the development of connected vehicles, a first revolution in the way information is produced and distributed to vehicles has arisen. Namely, vehicles became not only consumers of information produced by the infrastructure, but started producing their own information to be exchanged directly with other vehicles. Vehicles were equipped with on-board sensors which were used to generate safety notifications to be forwarded to other connected vehicles. In turn, standards safety messages such as the European Cooperative Awareness Message (CAM) or the American Basic Safety Message (BSM) were introduced for vehicles to communicate information

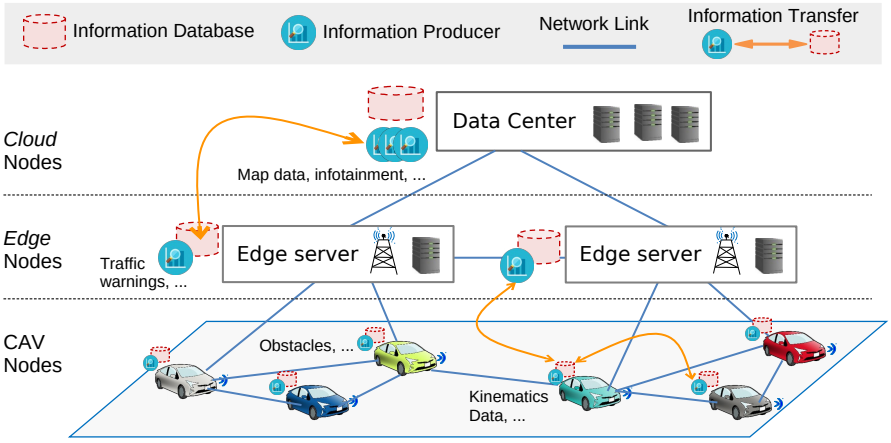


Figure 1.1: Node and Information Architecture in Vehicular Networks

about their position, speed, expected trajectory, and further kinematics data. In turn, Figure 1.1, illustrates the architecture of information distribution in vehicular networks, between vehicles, infrastructure units, and other connected vehicles. The CAM specification is further detailed in Section 1.1.2.

While the initial safety messages allowed connected vehicles to be aware of other connected vehicles and their intentions, they do not include legacy vehicles and vulnerable road users such as pedestrians. As such, an obstacle which was sensed by a connected vehicle might not have been detected by another and constitute a safety risk. In turn, mechanisms were defined to exchange information about obstacles sensed by a connected vehicle with other vehicles. For example, the Collective Perception Message (CPM) is a European specification being standardized, which lets connected vehicles exchange their object sensing to others. The CPM specification is further detailed in Section 1.1.2.

As such, the first revolution to information exchange in vehicular networks was the ability for vehicles to produce information themselves and communicate it to other connected vehicles. To support safety applications, vehicles share information which they sense with other connected vehicles to warn them of obstacles or safety challenges they might not have sensed themselves. Nonetheless, once a connected vehicle has been warned of an incoming obstacle, how should it react? Reacting to unexpected situations and avoiding obstacles requires experience from the vehicle, and the building of a form of complex information processing which we call *knowledge*.

Knowledge can be applied to many aspects of the fast-evolving vehicular networks:

- To begin with, knowledge or experience can be used to optimize vehicular communications themselves, by dynamically allocating network resources according to vehicular mobility.

- What is more, knowledge can be used to abstract a larger set of information and convey the core understanding behind it, to avoid redundancy. For example, what if, instead of requesting large amounts of information about the

behavior of human drivers in a new city, a connected vehicle could directly request for *knowledge* on how to drive safely in this new city? The knowledge would have been obtained through experience and analysis of larger amounts of human driver behavior information.

- Then, knowledge can be used to support complex vehicular applications through its ability to convey experience and abstractions obtained from the analysis of information. For example, highly-automated driving relies on decision-making based on knowledge and models of the environment which have been trained to recognize the behavioral patterns of other road users, as well as detect whether specific objects are dangerous obstacles or not.

As such, the generalization of the use of knowledge, in addition to information, in vehicular networks constitutes a second revolution in the way content is treated by vehicles. Vehicles are transitioning from information consumers and producers to *intelligent* vehicles which make use of knowledge obtained from experience, to eventually reach goals such as highly automated driving. Knowledge plays an increasingly important role in the software and functioning of connected vehicles, to power decision-making algorithms used for example in navigation.

What is more, as connected vehicles are ultimately responsible for their own decisions in applications of highly-automated driving, they require their own knowledge which is adapted to their experience and their specific context of driving. So far, each automaker is responsible for its own knowledge building, to be used by vehicles in their fleet. Locally, each vehicle is responsible for tailoring the knowledge to its context, based on its own experience. Yet, is it really optimal for each vehicle and each automaker organization to recreate their own knowledge from scratch to address the same challenges? Instead, what if connected vehicles could communicate their knowledge to other vehicles, as they can communicate their information? What if connected vehicles could exchange knowledge models to perform automated driving in a specific area, rather than relearning the same knowledge from information? Finally, what if connected vehicles could cooperate to build new knowledge which benefits all vehicles? Existing information sharing mechanisms must be extended to support the exchange of *knowledge* and allow connected vehicles to communicate and cooperatively build their knowledge, to be reused by others.

In turn, this doctoral work considers and investigates the problematic: What are the characteristics of vehicular knowledge and what are the requirements in terms of knowledge description to transition from information-centric to knowledge-centric networking in vehicular networks, providing the support of context-aware distributed knowledge creation, composition and application?

1.1.2 Vehicular Networks

Various standards and technologies already exist and have been developed for the storage and networking of information in vehicular networks, either from vehicle to vehicle or from vehicle to infrastructure.

Information Storage

Information can be stored on-board connected vehicles through standardized databases, such as the ETSI *Local Dynamic Map (LDM)* information base [4]. The LDM is divided into four layers, namely (i) permanent static data, i.e., map data, (ii) temporary static data, i.e., roadside infrastructure, (iii) temporary dynamic data, e.g., roadblock, signal phase, and (iv) highly dynamic data, e.g., vehicles, pedestrians. Due to the limited storage space on board connected vehicles, the LDM involves content maintenance mechanisms which discard information which has timed out or is not spatially relevant anymore.

What is more, to ensure interoperability of information for critical domains such as safety, standards have been defined to name and store information in a mutually-understandable manner. As introduced in Section 1.1.1, safety messages have been standardized both in Europe and the USA, which describe vehicle kinematics and road events as a formal basis for more elaborated applications. The CAM, standardized by ETSI in [5], equivalent to the BSM in American standards, contains information about the current position, velocity and expected trajectory of the ego vehicle. Decentralized Environmental Notification Messages (DENM), as standardized by ETSI in [6], can be used to encode and notify drivers of various abnormal road events including accidents, roadwork, heavy precipitation or traffic congestion. What is more, CPMs are being standardized by ETSI and described in [7]. They allow a connected vehicle to share information about a sensed object to other vehicles on the road. CPMs have the potential to allow automated vehicles to cooperatively visualize their environment, including legacy human-driven vehicles, as considered in [8].

Information Networking

Vehicular networks involve connected vehicles and infrastructure nodes. Infrastructure nodes may consist of distant centralized *cloud* services which produce and provide information to vehicles. Alternatively, infrastructure nodes may be placed closer to the vehicles. As such, edge computing units provide computation capabilities as well as caching mechanisms closer to the road, to reduce the delay for providing information to vehicles. As illustrated by Figure 1.1, information is produced both by infrastructure nodes, and directly by connected vehicles. In turn, protocols and architectures have been defined to support the communication and networking of information between nodes of vehicular networks, which we refer to as Vehicle To Everything (V2X) communications. In this section, we introduce *the Internet of Things* and *Information-Centric Networking*, which have been used as two alternative networking paradigms to organize communications in vehicular networks.

Internet of Things Architecture The Internet of Things (IoT) is a concept in which an increasing number of real-world objects are provided with a network connection and interconnected such that nodes in IoT networks have access to content generated by other entities. It has applications in a variety of fields such as smart

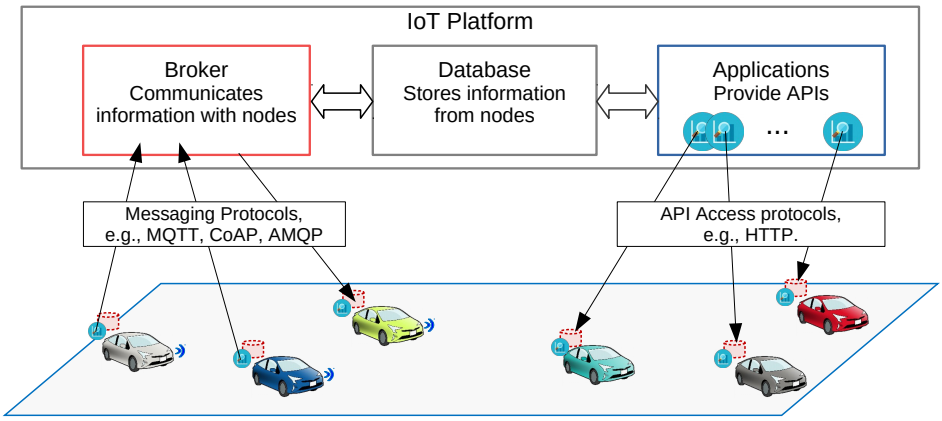


Figure 1.2: Vehicular IoT / IoV Architecture for Information Networking

cities, wearable technology, and notably vehicular networks. The concept of IoT is supported by an ecosystem of technologies and protocols to enable the distribution of the content produced by nodes. Typically, IoT content distribution is organized through application level protocols which run over a traditional Content Delivery Network (CDN) Internet architecture, where content is retrieved by sending requests to specifically addressed hosts, e.g., over TCP/IP. As illustrated by Figure 1.2, the nodes which produce information are connected with an *IoT Platform* host which receives incoming information from them through specific IoT messaging application level protocols. In turn, the IoT platform uses the information to support high-level services or applications that are provided to users using traditional HTTP-supported APIs. Applied to vehicular networks, IoT can be referred to as the Internet of Vehicles (IoV), where the considered nodes are connected vehicles, connected road users, or infrastructure units, as illustrated by Figure 1.1. Application level protocols which organize the communication between the IoT platform and nodes include *MQTT* [9], *CoAP* [10], or *AMQP* [11]. For example, *MQTT* is a publish&subscribe-based messaging protocols in which nodes can subscribe to user-defined *topics*. When another node pushes information to that *topic*, it is forwarded to nodes which have subscribed to it.

Information-Centric Networking Architecture Information-Centric Networking (ICN) is a candidate paradigm of information distribution which has been applied to vehicular networks, as surveyed by [12]. Traditionally, nodes and services of the Internet are interconnected following a Content Delivery Network (CDN) approach. Large data centers hold extensive amounts of information which can be downloaded by nodes provided the knowledge of a host address where the information can be fetched from. ICN uncouples data from its location. Rather than by host address, information is addressed by its name. Combined with information caching and routing mechanisms, ICN allows to alleviate the network load on traditional data centers. Information-centric networks are composed of three main building blocks: information routing, caching, and naming. ICN routing schemes should efficiently route information requests to their respective publishers. Caching mechanisms are a key to reduce the number of hops required to retrieve a given piece of information. Finally, a naming scheme must be implemented to identify requested information items. Figure 1.3 compares the routing process of ICN and of traditional CDN

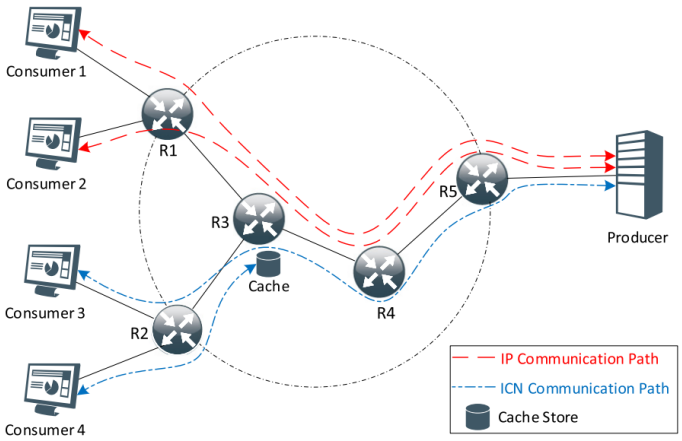


Figure 1.3: Comparison of CDN and ICN Content Retrieval [12]

information delivery. ICN has been studied to support information networking in realistic vehicular applications such as the distribution of safety warnings. In [13], vehicles sense and publish black ice warnings which are forwarded to subscribing vehicles through ICN.

Towards Knowledge Networking

Definition of Knowledge Information storage and networking mechanisms described as part of vehicular networks allow vehicles to exchange information such as infotainment, map data, or safety warnings, e.g., the detection of an obstacle on the road. Yet, as connected vehicles evolve to *intelligent* vehicles which ultimately target highly automated driving, reacting to safety warnings sent by other connected vehicles requires sophisticated decision-making based on *experience*.

Gradually, the complexity of the information processing operations performed by nodes of vehicular networks has increased to support complex smart applications such as platooning or highly-automated driving. Such applications do not rely on simple information processing and presentation to the user or driver. Instead, the information which is obtained by sensor output or received from other vehicles is used to produce abstract *knowledge* about the environment, which is used to support decision-making. As such, we witnessed a shift from information-based applications to *knowledge-based* applications in vehicular networks.

Knowledge has been used to refer to abstract content and models produced from input information. As illustrated by Figure 1.4, knowledge is built through the analysis of information, which is itself based on the processing of data. Data is defined as an atomic value with a unit, e.g., $30kph$. Information is built by aggregating pieces of data that describe a situation. For example, $(17:00, 30kph)$ refers to a vehicle's speed at a given time. Lastly, *knowledge* describes general patterns and relationships obtained through the analysis of sets of information. For example, [*If the mean speed of vehicles is 30kph at 17:00 then the school rush hour has started*] is knowledge associating a time and speed information with a context, i.e., the school rush hour. *Knowledge* has been used to refer to both (i) models produced from large sets of input information, able to produce an abstract output,

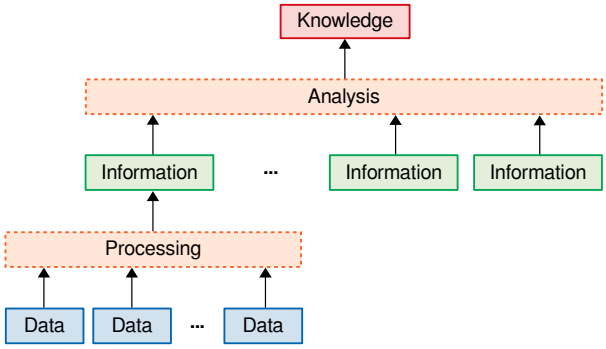


Figure 1.4: The Relationship between Data, Information and Knowledge

and (ii) the abstract output itself produced by knowledge models, as it is an abstract content yet similar in structure to information.

Vehicular Knowledge Building Artificial Intelligence (AI) is a broad class of approaches and algorithms designed to create and represent *knowledge*. It can be divided into several branches, including among others mathematical optimization, reasoning based on logic symbols, or Machine Learning (ML). ML is a popular domain of AI which is itself divided into three broad classes, i.e., supervised, unsupervised, and reinforcement learning, as illustrated by Figure 1.5:

1. Supervised learning is applied to classification or regression operations. It is a learning-by-example approach. Namely, a model is *trained* based on a number of samples of the form: (information, class) for classification – or (information, value) for regression. Provided with a sufficiently large set of input examples, the model is able to accurately predict the class or the value associated with an unseen example. Once the model reaches a satisfactory threshold in accuracy, it is said to be *trained*. Knowledge is extracted as the relationship between the information and its associated class, i.e., the function that takes information as an input and returns its estimated class. Algorithms which can train supervised learning models include linear and logistic regression, decision trees, neural networks, support vector machines, or naive Bayes classifiers.

2. Unsupervised learning is designed to identify relationships and patterns between objects in a dataset which have not been manually annotated with labels. Instead, patterns are exposed through the extraction of clusters of similar items in the provided input dataset of information. As such, knowledge is created by exposing potential relationships among information items and sorting them into different clusters. Algorithms implementing unsupervised learning include K-Means, K-Nearest neighbors, or hierarchical clustering.

3. Reinforcement learning can be used by an agent to learn the optimal behavior to adopt in a context of interaction with an environment to maximize a user-defined reward. Reinforcement algorithms aim to learn a *policy* to choose the action that maximizes the received reward, based on the current state of the environment. Q-learning is a popular example of a reinforcement learning algorithm.

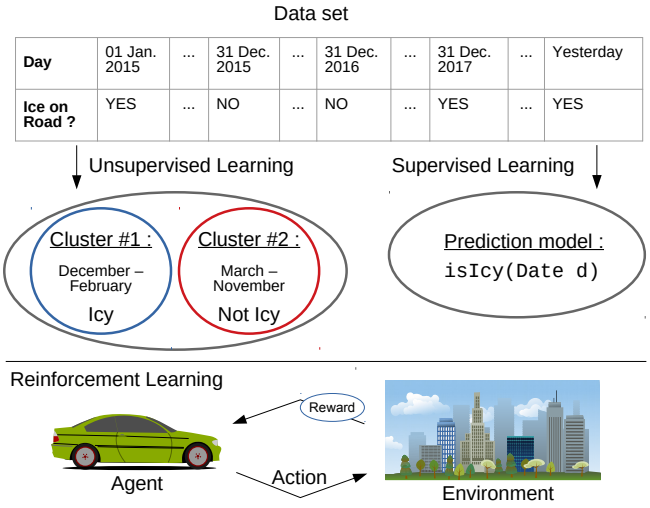


Figure 1.5: Supervised, Unsupervised and Reinforcement Learning

AI algorithms can produce *knowledge*, in turn supporting smart applications which require complex processing. As introduced in Section 1.1.1, information exchange mechanisms notably allow the transmission of safety notifications between connected vehicles, e.g, using the CPM specification. Yet, provided with the information that an obstacle was detected ahead of a connected vehicle v, the reaction of v to the obstacle, e.g., in a case of highly-automated driving, requires experience and complex processing, i.e., *knowledge*. As a follow-up on the use case of decision-making based on the distribution of safety notifications by connected vehicles, we quote and describe existing works which produce *knowledge* from simple environmental observations and *experience*, in order to react to obstacle warnings for automated driving.

Knowledge has been trained in existing works to allow Highly-Automated Vehicles (HAVs) to perform automated maneuvers to avoid an obstacle, either static or moving. In [14], Reinforcement learning is used to train a complex obstacle avoidance behavior. In this case, the knowledge training process involves attempting many approaches to avoid the considered obstacle, which are rated based on the level of success of the maneuver, e.g., collisions are highly penalized. After a sufficient amount of training iterations, through *experience*, a knowledge model has been learned which allows vehicles to efficiently avoid collisions in complex situations. Alternatively, an approach to produce obstacle avoidance knowledge based on a mathematical optimization algorithm is described in [15].

Knowledge Networking Generally, knowledge as the extraction of potentially complex and abstract understanding from input static information is a key to implement novel smart applications. Regardless of the technique, extracting knowledge from information is a complex and expensive process, as investigated in [16]. So far each vehicle remains autonomous for its knowledge building, which requires highly specialized algorithms and a large amount of input information, potentially sourced from multiple different vehicles. Yet, the generated knowledge may be beneficial to other vehicles. This can be seen as a significant overhead considering that knowledge

can be shared and not individually recreated.

Nonetheless, knowledge is a highly contextual content, which requires remote nodes to mutually understand the context in which a given knowledge model should be applied or further trained, in order to be shared. For example, a piece of knowledge trained to avoid obstacles, e.g., other vehicles as they behave in Californian 4-way stop intersections, might not be adapted to run in a roundabout in a country where vehicles drive left. As such, it is critical to accurately convey the context associated with the application or the training of a knowledge model to allow interoperable knowledge creation, composition and application. Such contextual annotations are keys to implement efficient and interoperable knowledge networking. As such, the requirements for knowledge sharing include:

1. A description of knowledge models, including their interface, i.e., inputs and outputs, as well as a description of their context of application or training. The aim of the model description is to make knowledge mutually understandable, and to be able to use a knowledge model which was trained by a remote node.

2. An integration of knowledge description to knowledge storage, routing and sharing mechanisms, in order to provide vehicles with the right piece of knowledge for the right context.

What is more, the definition of knowledge sharing mechanisms in vehicular networks should take into account the larger range of operations which can be associated with knowledge, as opposed to information. Namely, information sharing mechanisms involve the distribution of copies of the information to remote nodes. While copies of trained knowledge can be shared with other nodes of the vehicular networks, an operation of knowledge is to request the direct creation of knowledge in a remote node, without retrieving copies of the knowledge model used to produce it. We can call this operation *knowledge as a service*, as the model is created in a remote node, on behalf of a requesting node. For example, Named Function Networking (NFN) is a technology to request the in-network application of a function to directly retrieve its output [17].

Additionally, a key operation of knowledge distribution is the cooperative training of a model by several nodes in a distributed manner. Namely, distributed ML model training algorithms have been developed, such as distributed ML or federated ML. In the latter, each training node trains a fraction of the considered model, before they are gathered to iterate the training, as illustrated by Figure 1.6. As a consequence, like for the sharing of a trained piece of knowledge, models which are being cooperatively trained by distributed nodes must be described by a common vocabulary or language, to let each node understand how to produce model updates for the considered model.

Information sharing has been widely investigated and even standardized. Yet, no mechanisms have been defined to combine a semantic contextual description of knowledge with cooperative training or application mechanisms in vehicular networks, such that vehicles can request a piece of knowledge which matches a specific context, or request the application of a specific model in a given context. So far, existing information networking mechanisms lack semantic annotations describing

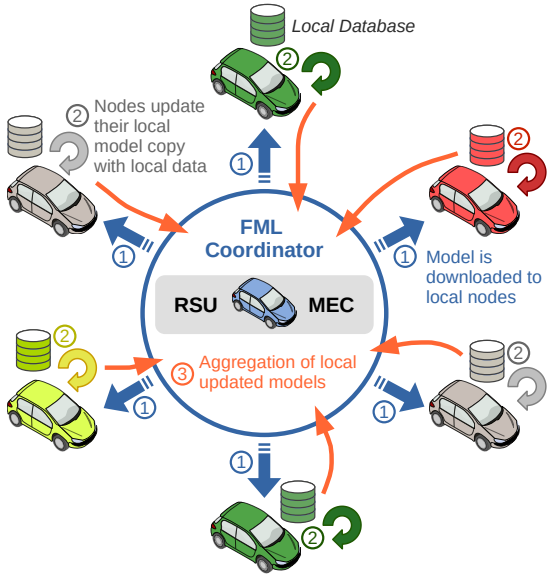


Figure 1.6: An Iteration of Federated Learning Training

such context, which prevents them to uniquely locate and acquire the required knowledge. The naming, storage, and dissemination of knowledge thus must integrate such annotations, which in turn need to be harmonized.

1.2 Methodology

To address the aforementioned problematic, we define a framework to extend existing information-centric mechanisms and support knowledge distribution in vehicular networks, which we name *Vehicular Knowledge Networking* (VKN). Namely, as part of VKN, we describe a set of concepts for knowledge distribution, which are divided into three complementary blocks: (i) knowledge description, (ii) knowledge storage, and (iii) knowledge networking. In turn, we implement aspects of the framework in various work items to demonstrate the feasibility and the performance of knowledge distribution through VKN.

Knowledge Description To begin with, we consider the problematic of the description of knowledge to make it mutually understandable among nodes of vehicular networks. A first aspect of knowledge description is: What is the nature of the knowledge? Where does it come from? What is the set of inputs it requires and what are the outputs it produces? Such description requires a common vocabulary, and uniquely named items to refer to the input and output interface of the knowledge. In turn, another aspect is: how and where can the knowledge be used, i.e., in which context? Similarly, a vocabulary must be defined to let vehicles describe a specific context of training or application of a knowledge model, to make sure that it is used in the right context.

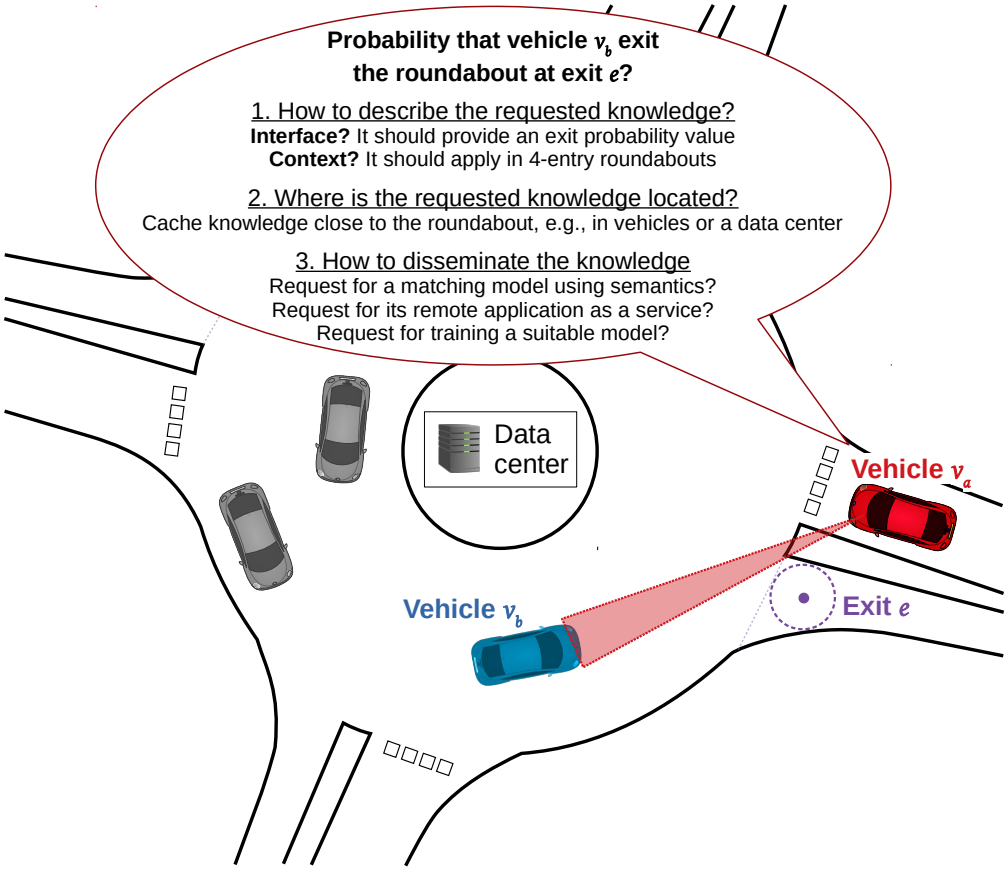


Figure 1.7: Example Vehicular Knowledge Networking Case Study

Knowledge Storage In turn, we investigate on the impact of the definition of knowledge description semantics on the storage of knowledge in nodes of vehicular networks, i.e., both connected vehicles and infrastructure units. Namely, we consider whether the storage of knowledge requires a change and has a significant impact on the existing mechanisms of information storage in vehicular networks. For example, can knowledge be stored at the same level than information, in LDM information bases on board vehicles?

Knowledge Networking Finally, based on knowledge description and storage considerations, we investigate to what extent knowledge can be networked in vehicular networks, and whether novel protocols must be designed to support knowledge networking. What is more, we investigate whether knowledge networking requires a significant change in the protocol stack which supports information networking, or if it can be implemented as an overlay.

These three aspects of vehicular knowledge networking are complementary and interconnected. For example, let us consider a use case of knowledge trained to react to an obstacle which was detected on the road, as considered in Section 1.1.1. Namely, we consider a knowledge model which estimates the probability of a vehicle detected on a roundabout to leave at the next exit. As illustrated by Figure 1.7, the following mechanisms need to be defined in vehicular networks:

- A semantic description of the knowledge, as well as of its context of application, e.g., the type of roundabouts it applies to, should be provided so that vehicles know how to use the knowledge to get accurate results.

- In turn, the semantic description is leveraged by vehicles and infrastructure nodes to cache copies of the knowledge models in relevant areas. For example, vehicles which are about to cross a roundabout in a given context could request a relevant knowledge model to be cached in an infrastructure node close to the roundabout.

- Then, knowledge networking mechanisms must be defined to let vehicles request for the caching of the knowledge, as well as other operations, such as the remote creation of exit probability knowledge, or its cooperative training for new roundabouts. What is more, such requests must be expressed using the vocabulary described as part of the semantic description of the roundabout exit probability knowledge model.

In this doctoral work, we define a conceptual framework considering all three aspects of knowledge distribution in vehicular networks. In turn, we consider each part of the framework to evaluate its performance for knowledge distribution using various metrics. In particular, we focus on the operations of *cooperative knowledge application*, i.e., when fully trained knowledge is shared, and *cooperative knowledge training*, i.e, when knowledge is trained cooperatively by several nodes. In light of the definition and evaluation of knowledge distribution aspects in vehicular networks, we specifically consider the following open questions:

- Does knowledge networking require a significant change in networking protocols at the networking or transport layers, or can it be implemented as an overlay to existing information exchange architectures?

- How critical is the definition by model makers of a description of the context of training and application of their models for the accuracy of knowledge distribution in vehicular networks?

1.3 Contribution

The contributions of this doctoral work to the literature are as follows:

- A conceptual framework for cooperative knowledge (i) description, (ii) storage, and (iii) distribution in vehicular networks is defined. The framework has been the subject of a magazine article publication in [18]. This item is transversal to the knowledge description, storage, and networking part of the methodology. Namely, we describe the requirements for a semantic description of knowledge which makes operations such as cooperative knowledge application and training in vehicular networks possible. In turn, standards which match these requirements are described and exemplified. Furthermore, we describe a knowledge layer between the application layer and the information

storage layer, containing the LDM, on-board connected vehicles and nodes of vehicular networks. Finally, we describe the requirements for the expression of knowledge requests between nodes of the vehicular networks, which let vehicles synchronize to exchange trained knowledge, request remote knowledge creation, or organize to set up a cooperative ML knowledge training process.

- The proposed VKN framework is evaluated theoretically and shown to enable overhead reduction through cooperative knowledge application, as well as increased training speed and accuracy for cooperative knowledge learning, through the orchestration of federated learning. This item is transversal to the knowledge description, storage, and networking part of the methodology. It has been the subject of a conference publication in [19]. Namely, the conceptual description of the VKN framework as described in Item 1.3 is partly implemented to demonstrate its impact and potential performance improvements for operations of (i) *cooperative knowledge application* and (ii) *cooperative knowledge training*.

- A study on the risk-based routing of HAVs is performed, which was presented at MFTS 2020, and published as a research report in [20]. This item is a prerequisite to the knowledge description and networking aspects of the methodology, as it shows the need for risk knowledge to be created, described, and networked to vehicles. It shows that a naive routing of HAVs away from areas which have been set as *risky* for HAVs, based on their shape and characteristics, deteriorates traffic conditions and increases emissions in a simulated city of Monaco. In turn, this illustrates the need for a piece of knowledge to dynamically assess risk in various areas, such that vehicles can reduce driving risk, while not strictly avoiding certain areas, which impacts traffic and the environment.

- In turn, novel knowledge to estimate driving risk on a roundabout is trained. This item involves the knowledge creation and description part of the methodology. To begin with, knowledge models are trained which are able to predict the probability of a vehicle to exit a roundabout at the next available exit, based on real vehicle track data extracted from drone-captured datasets. This knowledge can be used to support the safe crossing of a roundabout, e.g., when a vehicle which is about to enter computes the probability of an incoming vehicle to exit. What is more, an estimation of the level of risk in a roundabout, based on measured time to collision values is computed. The production of roundabout driving risk knowledge as (i) the exit probability and (ii) the time to collision values of vehicles has been the subject of a conference publication in [21].

- Parallelly to the definition of roundabout exit probability knowledge, we perform an analysis to determine a semantic description of the context of training and the context of application of exit probability models. This analysis will be submitted to the 'Transportation Research Part C' journal, as listed in [22]. This item involves the knowledge description aspect of the methodology. Namely, the roundabout semantic characteristics, i.e., roundabout context data, which impact the similarity of two distinct exit probability models are identified. In turn, we show that an exit probability model performs more

accurately when it is applied to a context which is similar to its context of training, compared to models which were trained in different contexts. What is more, the training of an exit probability model for a new roundabout is more accurate when the available training data is completed using training data extracted from a known roundabout featuring a similar context. As such, our contribution shows that the semantic description of the context of application of a knowledge model improves the accuracy of knowledge distribution in vehicular networks. We argue that model makers should perform a similar mutually understandable context description for other models which are highly dependent on context.

- We simulate the context-aware *cooperative knowledge application* as described in Item 1.3 through a packet-level simulation in which vehicles which enter a roundabout request the creation of exit probability knowledge to an infrastructure unit, based on the context of the roundabout. This item involves the knowledge storage and networking aspects of the methodology. It has been submitted to the WONS 2022 conference, as listed in [23], and involves two contributions. On the one hand, we demonstrate accuracy improvements related to context-aware knowledge creation while maintaining low delay and overhead in a realistic use case of vehicles requesting roundabout exit probability knowledge creation, compared to a traditional approach which does not take the context of application of the knowledge into account. On the other hand, we perform an implementation of VKN involving the description, storage, and sharing of the exit probability knowledge to perform the knowledge distribution in the packet-level simulation. The implementation acts as an overlay of a standard NDN architecture. The obtained results indicate that existing protocols for information sharing in vehicular networks can support knowledge distribution without significant changes at the networking or transport layers. Instead, application protocols which integrate knowledge description, storage, and sharing operations can support knowledge distribution in vehicular networks.

1.4 Structure of the Thesis

This thesis manuscript is divided into several chapters, which we briefly introduce and describe in this section.

To begin with, Chapter 2 summarizes the state-of-the-art literature for knowledge networking in vehicular networks. On the one hand, a conceptual definition of knowledge as opposed to information is described from existing theoretical works. Moreover, the literature on all major algorithms and techniques of knowledge creation and representation in computerized systems is reviewed. On the other hand, the existing mechanisms for content storage and caching in vehicular networks, as well as content distribution, are introduced. In turn, the introduction of the literature on both the definition of knowledge as a content and the means of content distribution in vehicular networks allows to confront the two aspects and identify the limitations of knowledge distribution in vehicular networks, to clarify the problem to be addressed in this doctoral work.

Based on these descriptions, Chapter 3 analyzes the set of operations related to knowledge networking in vehicular networks as opposed to information networking. Namely, information networking standards have been defined to distribute copies of uniquely named content. Yet, knowledge networking involves additional operations such as cooperative knowledge creation, composition, and application, which in turn requires a mutual understanding of knowledge. This chapter analyses the limitations of current information-based content distribution mechanisms in terms of their ability to convey and take into account the strong contextual meaning which is associated with knowledge, especially in operations which require synchronization between nodes such as cooperative knowledge creation.

Chapter 4 introduces a conceptual framework to integrate knowledge description mechanisms, including a description of the context required to handle a piece of knowledge, to vehicular networks. This knowledge description integration is used to describe cooperative knowledge creation, storage and distribution operations in vehicular networks. We name the framework *Vehicular Knowledge Networking*. In turn, evaluations of theoretical performance increases in terms of overhead, and cooperative knowledge training speed and accuracy, are performed which show performance improvements.

Chapter 5 introduces and describes a real case of knowledge creation, namely, the creation of driving risk knowledge in roundabouts by vehicles. This application is supported by realistic data and aims at defining a concrete application to implement and evaluate VKN with. First, a preliminary study illustrates the need for risk knowledge to be created to improve the navigation of connected and autonomous vehicles in cities. Then, real roundabout data is used to produce risk knowledge in roundabouts.

Chapter 6 uses the roundabout risk knowledge created in Chapter 5 to implement and evaluate knowledge distribution through VKN. First, roundabout knowledge, as the probability of a vehicle to exit at the next available exit in a roundabout, is produced for a set of multiple roundabouts. A description of the context of each knowledge model, as well as an analysis of the impact of the context of training of the roundabout knowledge on its accuracy is performed. In turn, the produced knowledge description is used along with an implementation of the VKN framework to network accurate exit probability knowledge to vehicles at the entry of a roundabout.

Finally, Chapter 7 derives conclusions from the results of Chapter 6, supported by the preliminary work of Chapters 4 and 5. What is more, perspectives for the future implementations of knowledge distribution in vehicular networks, as well as their relationship with the definition of context semantics adapted for each knowledge, are discussed. At last, the list of the publications associated with this doctoral work is provided.

Chapter 2

State of the Art

In this chapter, we describe the state of the art at the time of writing of various content distribution approaches in vehicular networks and their limitations when applied to knowledge distribution. Generally, we review the state-of-the-art of the major fields composing a vehicular knowledge network.

2.1 Definition of Vehicular Networks

Vehicular networks involve the interconnection of objects in the vehicular environment. They include:

- Connected vehicles and road users, e.g., pedestrians with a smartphone.
- Road infrastructure nodes, e.g., road side units or connected traffic lights.
- Other infrastructure units, such as data centers or edge servers, i.e., computing and storage units close to the road.

2.1.1 Motivation for Vehicular Information Networking

In early applications of *connected vehicles*, the need for vehicular networks was expressed as a tool to support scenarios to improve: (i) safety, (ii) traffic efficiency and (iii) passenger infotainment, as described by the original manifesto of the CAR 2 CAR communication consortium [24]. As illustrated by Figure 2.1, the interconnection of nodes in a vehicular network would allow (i) drivers to receive safety notification or route recommendations, e.g., from other vehicles or connected traffic lights, (ii) traffic operators to receive traffic information from vehicles, to take educated decisions to optimize the traffic flow, and (iii) connected objects, such as gas stations, to communicate relevant information in advance to drivers, e.g., the price of gas.

For example, *TrafficView* [25] is an early vehicular networking application to assess traffic conditions on the road ahead of a connected vehicle. Each vehicle

Figure 2.1: Overview of the Initial Scenarios Enabled by Vehicular Information Networking [24]

wirelessly transmits information about its kinematics to neighboring vehicles. In turn, vehicles maintain a dynamic map of their neighboring vehicles and can present the driver with dynamic traffic conditions up to several hundred meters ahead.

What is more, *Cooperative Collision Warning* (CCA) is a class of systems to provide timely safety information to drivers to avoid crashes. As described in [26], a human driver alone might in some cases fail to notice a hard-breaking event of a front vehicle if it is obstructed, e.g., by fog or another vehicle. CCA systems enable the exchange of wireless safety notifications between vehicles, such that even a hidden hard-breaking event can be notified timely to drivers in order to avoid a potential collision.

Such scenarios and applications have motivated the need for the definition of protocols adapted to the networking of information among vehicles, infrastructure, and connected objects on the side of the road. Yet, the vehicular environment comes with a specific set of challenges which must be addressed in order to efficiently implement the networking of information, in Vehicle To Vehicle (V2V) and Vehicle To Infrastructure (V2I) cases, summarized as Vehicle to Everything (V2X). In the next section, we provide an overview of the challenges related to the implementation of V2X networking, caused by the peculiarities of the vehicular environment.

2.1.2 Vehicular Networking Challenges

The vehicular environment involves specific challenges for the implementation of wireless networking among vehicles and infrastructure, which were surveyed in [27]. A main characteristic of vehicular networks is the high mobility of vehicles, leading to potentially large disparities in the relative speed between nodes. As such, direct communications links between vehicles and infrastructure nodes are typically short-lived, as nodes may quickly move out of communicate range from each other. Namely, the formation and loss of communication links between vehicles is common, and the channel link quality, potentially unstable. As such, a challenge of 'horizontal' vehicular wireless networking, i.e., between vehicles and other vehicles or road-side infrastructure nodes, is the support of multi-hop forwarding algorithms

for data packets, and the formation of routes which need to be recomputed often.

As such, vehicular mobility is a main aspect of vehicular networks, which requires specific wireless networking stacks which are adapted to these conditions. What is more, the high volatility of the communication links among vehicles is accompanied with vehicular smart applications which require real-time or low delay delivery of information between vehicles to support safety applications. For example, as described in [26], CCA systems require the dissemination of safety notifications about a hidden hard-break event in near real-time. As such, vehicular networking protocols must enable the low delay delivery of information between vehicular nodes, while taking the challenges of the high mobility of vehicles into account.

Specifically, [27] lists the following challenges for the implementation of vehicular networking:

Addressing & Geographical Addressing In some vehicular applications, the address of nodes is required to be linked to their physical position, to allow routing content in a single relevant area. For example, an application which requires geographical addressing is the dissemination of DENM [6] safety warning messages to a specific location to warn vehicles about a road even such as an accident which happened in this location. Given the high mobility of vehicles, the implementation of geographical addressing is highly challenging, as it requires frequent updates.

Delay & Packet Prioritization As introduced earlier, some vehicular applications may require the transmission of information packets with very low latency, such as applications of safety warning dissemination. Yet, all vehicular applications do not come with strict delay constraints, such as, for example, the retrieval of multimedia content for passenger infotainment. As such, mechanisms of prioritization of some data packets are required to ensure the fast delivery of safety notifications even in cases of high channel utilization.

Reliable Transport It is challenging to provide a stable transport service for information dissemination over an unreliable vehicular network where existing routes often break due to the high vehicular mobility. In turn, cross-layer approaches which integrate the networking and transport layers are adapted for vehicular networks.

Trust, Privacy & Security Security protocols should be implemented to ensure the reliability of information which is exchanged among vehicular networks. Namely, mechanisms should be defined to trust the entity which produced a piece of information, or verify its integrity. Simultaneously, the privacy of the identity of the information producer should be maintained, to comply with legal requirements in certain countries.

2.1.3 Existing Protocol Stacks

Several protocols have been standardized to implement vehicular networking, given the specific challenges described in Section 2.1.2. In this section, we perform an

overview of the protocol stacks of the two main existing architectures for the specification of V2X communications. Mainly, two architectures have been standardized, i.e., the ETSI standardized Intelligent Transport Systems (ITS), and an architecture which combines Dedicated Short Range Communications (DSRC), the IEEE WAVE 1609.x protocols, and SAE standards for messaging. What is more, the physical and data link level technologies have been specified for two distinct approaches, i.e., one WiFi 802.11p/802.11bd-based approach, and one cellular-based C-V2X architecture. In this section, we present the existing protocols of the aforementioned architectures from the applications to the lower communication layers.

Applications

A roadmap and a set of use cases for vehicular networking applications have been defined as part of the ETSI ITS Basic Set of Applications (BSA) [28].

As described in [29], the applications are divided into:

- *Day 1* applications for 'awareness driving', in which vehicles are aware of other connected objects. Example applications include collision avoidance through safety notifications, and information retrieval from the road side infrastructure.
- *Day 2* applications for 'sensing driving' aim at letting vehicles sense and share information about non-connected objects to increase awareness. Example applications are the protection of vulnerable road users and semi-automated driving through collective sensing.
- *Day 3+* applications for 'cooperative driving' aim at letting vehicles cooperatively achieve automated driving, through, for example, the cooperative planning of complex driving maneuvers.

Similarly, the SAE J3016 standard defined several levels of automation for self-driving vehicles, from 0 for no automation to 5 for full automation without human assistance, which is the eventual goal for vehicular applications.

Messages

A set of messages have been defined as part of ETSI ITS and SAE standards to let vehicles exchange specific information items in an interoperable format to achieve the aforementioned applications. In this section, we describe a subset of the key messages to convey vehicular information in vehicular networks, organized by the 'day' they are associated to, as described in [29].

For *Day 1* applications, as part of the ETSI ITS standards, the CAM and DENM messages have been specified, respectively, in [5] and [6]. The CAM message lets vehicles exchange information about their kinematics, e.g., position, speed, and expected trajectory. The DENM message is distributed to inform vehicles of events of interest such as accidents, roadwork, heavy precipitation or traffic congestion. On the other hand, SAE standard messages include the Basic Safety Message (BSM)

and the Personal Safety Message (PSM), specified in the SAE J2735 standard [30]. The BSM is equivalent to the ETSI CAM, and the PSM is designed to be sent by users carrying mobile devices, such as pedestrians, to signal their presence to connected vehicles.

For *Day 2* applications, the CPM message is being standardized by ETSI in [7], to let vehicles share their sensing with other vehicles. Namely, using CAMs, connected vehicles are aware of the other connected vehicles, but cannot share information about non-connected obstacles or vulnerable road users. As such, the CPM lets connected vehicles describe an object or obstacle which was sensed and share the information with other vehicles, for cooperative sensing applications.

Stack

In turn, networking and transport protocols have been defined as part of the ETSI ITS and WAVE architectures to support the dissemination of the aforementioned messages in vehicular networks, considering the constraints of high-mobility and low delay safety applications of vehicles.

ETSI ITS ITS defines the Basic Transport Protocol (BTP) [31], running over the GeoNetworking [32] networking protocol. GeoNetworking specifies a routing of V2X messages based on the geographical position of nodes. As described in [33], it introduces the *GeoAnycast* and *GeoBroadcast* scenarios, i.e., respectively, communication to an arbitrary or to all nodes in a given geographical area. On top of GeoNetworking, BTP provides a minimal transport service, similar to UDP. Alternatively, ITS allows the use of TCP/IP or UDP/IP as transport/networking layers. What is more, IPv6 packets can be transmitted over GeoNetworking, as specified in [34]. Figure 2.3 illustrates the architecture of an ITS vehicular node.

IEEE WAVE The DSRC/WAVE vehicular networking architecture is based on the Wireless Access in Vehicular Environments (WAVE) protocol suite, which is specified in the IEEE 1609 family of standards. IEEE 1609.3 [35] is a cross-layer protocol of the networking and transport layers, which organizes the addressing of nodes and the routing of data packets among them. It supports either traditional TCP/IP and UDP/IP, or a Wave Short Message Protocol (WSMP), for 1-hop and typically broadcast V2X delay-critical safety messaging.

Technology

Two major alternatives have been developed for the physical and data link aspects of vehicular networking, to support the aforementioned information routing protocols: (i) WiFi-based approaches, using IEEE 802.11p and eventually its successor IEEE 802.11bd, and (ii) cellular approaches, using the C-V2X technology.

Wifi-based Two major architectures use WiFi as a basis for V2X communications. Namely, ITS-G5 in Europe and DSRC. Both architectures are based on the physical and MAC layer of IEEE 802.11p, and add functionality to the data link layer, e.g., IEEE 1609.4 to add multi-channel transmission capabilities to the MAC. In turn, this lets vehicular applications use specific channels to exchange delay-critical safety messages.

IEEE 802.11p is an amendment to the IEEE 802.11 WiFi standard which was designed to suit the vehicular environment, as specified in [36]. It specifies the physical and Medium Access Control (MAC) layers of the Open Systems Interconnection (OSI) model, for vehicular communications. One of the main characteristics of IEEE 802.11p is the definition of a method to exchange data without requiring prior authentication and association procedures, i.e., outside of the context of a Basic Service Set (BSS). As such, two nodes might start exchanging data as soon as they are in communication range, which is adapted to the high volatility of communication links among vehicular networks. Among the other notable modifications, changes are applied to support greater ranges and relative velocities than baseline IEEE 802.11. What is more, the upcoming successor of IEEE 802.11p is the IEEE 802.11bd standard, which is backward compatible with it, as specified in [37].

Cellular-based C-V2X architectures have been specified to support V2X communications over existing cellular communication protocols, as an alternative to WiFi-based approaches. As such, C-V2X includes LTE-V2X, specified in 3GPP Release 14 [38], which refers to V2X communications over LTE cellular networks, and 5G NR-V2X, specified in 3GPP Release 16 [39], which refers to V2X over 5G 'New Radio' radio access.

The service requirements for the support of V2X communications over LTE networks have been standardized by ETSI and 3GPP in [40]. Four modes of communication are defined, i.e., V2V, V2I, Vehicle To Pedestrian (V2P), and Vehicle To Network (V2N). V2N communications let vehicles operate wide-range communication over the traditional LTE Uu interface. On the other hand, V2V, V2I, and V2P communications use a specific LTE direct device-to-device interface named *PC5*. PC5 operates in the same spectrum bands than DSRC and ITS, and was optimized for high relative velocity differences. The C-V2X architecture specifies the physical and data link layers. As illustrated by Figure 2.3, IP or GeoNetworking protocols can be used on top of C-V2X.

2.2 From Information to Knowledge Networking

2.2.1 The Need for Knowledge in Vehicular Networks

V2X communication standards have allowed the exchange of information among vehicles, infrastructure, and other nodes of vehicular networks. Through communication protocol stacks which are designed for vehicular challenges such as high mobility, high relative velocity differences and unstable connection links, safety and infotainment messages can be exchanged among nodes of vehicular networks.

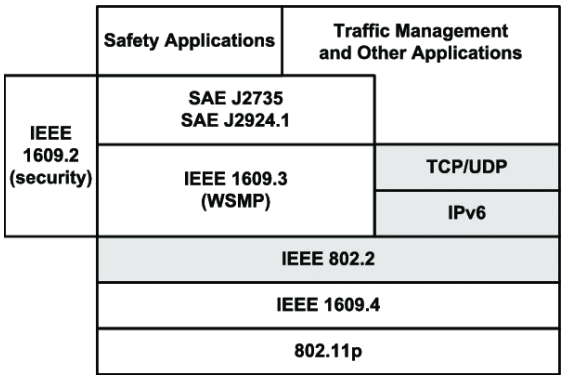


Figure 2.2: Protocol Stack of a DSRC/WAVE Node [41]

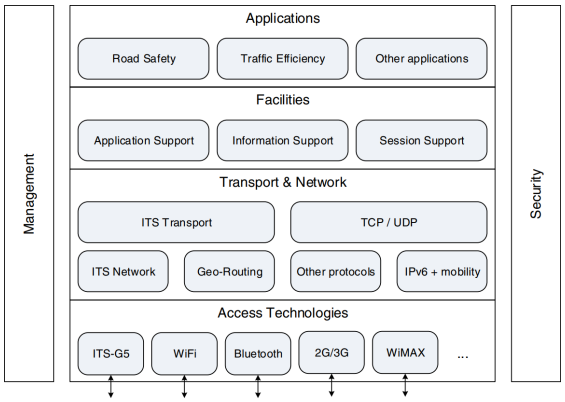


Figure 2.3: Architecture of an ETSI ITS Node [42]

For example, as described in [26], CCA systems can be supported by vehicular networking. Namely, safety notifications about a hidden hard-breaking event may be transmitted to relevant vehicles with a low delay, and presented to drivers. Similarly, to support modern vehicular applications, CPMs are being specified by ETSI in [7], for vehicles to exchange information about any obstacles which was sensed on the road. The information of the obstacle can be used to notify the driver of a danger.

Yet, in ambitious modern vehicular applications such as highly-automated driving, a connected vehicle is responsible for its own self-driving control. In turn, when a connected and highly automated vehicle receives an obstacle notification from another vehicle: How should it react? Does the obstacle require the execution of an evasive maneuver? If so, how should it be executed? Does it require coordination with other road users? Through smart vehicular applications, complex processing which is usually left to human cognition must be implemented by a machine. What is more, the challenges do not stop there. Even for non-automated vehicles, when an obstacle is detected, to which vehicle is this obstacle relevant, i.e., to which nodes should the notification be sent? When receiving several obstacle notifications, do the notifications actually represent the same obstacle?

These aspects, which are related to not only modern vehicular applications, but also the optimization of content distribution in vehicular networks, require a complex processing, a form of machine intelligence, which is typically referred to as *knowledge*. Artificial Intelligence (AI) allows the building of knowledge to be used in modern smart vehicular applications, and vehicular networking itself. It involves a broad body of techniques such as Machine Learning (ML), mathematical optimization, or expert systems. Generally, knowledge is taking an increasingly important place in vehicular applications and vehicular networks in general. Parallelly to the distribution of information through existing vehicular networking protocols, vehicular nodes extensively use and even create their own knowledge.

Table 2.1: Knowledge-Based Smart Applications in Vehicular Networks

Application	Ref.	Description	Technology
Knowledge-Based Networking Optimization	[43]	Resource allocation for vehicular wireless communications	Deep Learning
	[44]	Allocation of radio resources in a non-cooperative environment	Optimization
	[45]	Prediction of connectivity links duration for route choices	Machine Learning
	[46]	Reduction of the power consumption of vehicular communications	Federated Learning
	[47]	Prediction of wireless channel activity to schedule transmissions	Deep Learning
	[48]	Joint vehicular communication and mobility planning optimization	Reinforcement Learning
Knowledge-Based Storage & Computing Optimization	[49]	Probabilistic mobility-based information caching on the edge	Original Algorithm
	[50]	Social group learning-based edge content caching	Deep Learning
	[51]	Service caching location optimization for edge servers	Reinforcement Learning
	[52]	Information caching and computation offloading for delay optimization	Optimization
	[53]	Vehicle to vehicle computation offloading for load balancing	Reinforcement Learning
Knowledge-Based Vehicular Applications	[21]	Learning of exit behavior patterns of vehicles in a roundabout	Machine Learning
	[54]	Autonomous control of a vehicle, avoiding obstacles	Reinforcement Learning
	[55]	Optimization of the control of multiple traffic lights to minimize congestion	Reinforcement Learning
	[56]	Clustering of vehicle conflicts to infer risky areas	Machine Learning
	[57]	Predict the behavior of pedestrians, will they cross the road?	Machine Learning
	[58]	Road surface and friction estimation from front camera images	Machine Learning
	[59]	Near crash prediction based on vehicle kinematics	Machine Learning
	[60]	Object recognition from camera and LiDAR, e.g., vehicles, pedestrians	Machine Learning
	[61]	Learning of driver preferences to personalize forward collision alarms	Optimization
	[62]	Recognition of a high-level driving context	Expert Systems
	[63]	Collision risk assessment through the estimation of lane change probability	Original Algorithm
	[64]	Parking space availability prediction	Machine Learning
	[65]	Energy demand and driving range prediction of connected electric vehicles	Federated Learning

2.2.2 Existing Applications of Knowledge in Vehicular Networks

To demonstrate the increasing impact of knowledge in vehicular networks, Table 2.1 cites and classifies existing knowledge-based works in vehicular networks. Namely, we group knowledge application to vehicular networking in three categories: (i) knowledge-based networking optimization, (ii) knowledge-based storage & computing optimization, and (iii) knowledge-based vehicular applications.

First, in knowledge-based networking optimization, knowledge is built using AI algorithms and applied to the optimization of vehicular networking communications themselves. The optimization involves parameter choices at various layers of the communication protocol stack, e.g., low-level radio or wireless resources in [43, 44], or route definitions in [45]. The optimized parameters may be, for example, the power consumption, as in [46], or the reduction of packet collisions, as in [47]. Finally, the knowledge used for the optimization may use various approaches, such as the prediction of the duration of a connection link [45] or of the channel activity [47].

Then, in knowledge-based storage & computing optimization, knowledge is used to cache the information close to the vehicles which are likely to request it. Moreover, vehicles and edge units provide computation capabilities for vehicular nodes. In turn, knowledge has been used to optimize the distribution of computations. [49, 50, 51, 52] are examples of knowledge-based algorithms to optimize content caching in edge units on the side of the road. The optimized parameter is typically the delay of information retrieval. The knowledge can be, for example, based on the learning of mobility patterns [49] or social groups of vehicles [49]. On the other hand, [52,

53] optimize the offloading of computations in vehicular networks, e.g., to reduce computation delays [52] or balance the load among vehicles [53].

Finally, knowledge applies in a wide variety of smart vehicular applications, as non-exhaustively listed in Table 2.1. For example, a real vehicle is teaching itself to drive and avoid obstacles in [54]. In [55], a policy is learned to jointly control multiple traffic lights and reduce congestion.

2.2.3 Motivation for Vehicular Knowledge Networking

In some applications listed in Table 2.1, knowledge is produced using AI algorithms in a centralized data center, before being provided to vehicles, which then use it for smart applications. Yet, vehicles are increasingly involved in the knowledge building process. In some applications, just like they sense their own information, vehicles build or contribute in building their own knowledge. For example, in [46] vehicles contribute to build knowledge which allows a significant reduction of power consumption for ultra-reliable low-latency vehicular communications. In [65], vehicles cooperatively learn knowledge to better assess the energy demand and range of connected electric vehicles.

Then, similarly to Section 2.1.1 which motivated reasons to define vehicular networking of information produced by vehicular nodes, there is a need for a networking of *knowledge* in vehicular networks, so that vehicles can cooperatively build and exchange their own knowledge to support novel ambitious smart scenarios, as exemplified in Table 2.1. In turn, we present an overview of the forms that computerized knowledge can take in vehicular networks as a first step to identify the challenges of vehicular knowledge networking.

2.3 Knowledge Creation

2.3.1 A Generic Definition of Knowledge

The distinction between data, information, and knowledge is a nontrivial matter which has been investigated in the literature. In general, the definition of knowledge has been a debated topic among researchers and philosophers for centuries, if not millenaries. In antiquity, Plato defined knowledge as a 'justified true belief' [66]. Later, this definition of knowledge was further formalized under the name of the *tripartite definition of knowledge*, as detailed in Figure 2.4. Namely, while false propositions cannot be defined as knowledge, the justified beliefs obtained through analysis and abstraction of true facts match the general definition of knowledge.

This general definition of knowledge has been adapted to computerized information systems. A definition given in [68] is that knowledge is built from information, which is built from data. *Explicit knowledge* is what is left of generic and abstract knowledge when it is put on paper or machine code. Information is what is left of explicit knowledge when its subjective context has been taken away. Finally, data is information reduced to symbols, i.e., quantities with a measuring unit. For

The Tripartite Analysis of Knowledge: S knows that p iff

1. p is true;
2. S believes that p;
3. S is justified in believing that p.

Figure 2.4: The Tripartite Analysis of Knowledge [67]

example, when considering a set of spatial points sensed by a LiDAR, the set of positions of each point can be defined as data. Knowledge is the algorithm which is able to recognize objects from the set of point data, given a specific context, e.g., the current country or weather conditions. Finally, information is what is left when we remove the intelligence of context-based object extraction, and retain only the content produced by the knowledge, i.e., 'there is a vehicle ahead'.

An alternative definition is described in [69]. Data remains a symbol and information is defined as data which has been processed into a meaningful form for its user. It is also necessary to build knowledge. Finally, knowledge is information which has been understood by the user given its local context. The major difference between these two alternative definitions lays in which content can be considered knowledge. Knowledge as defined in [68] represents the algorithm able to produce abstract content, which is considered as information. On the other hand, the definition of [69] also considers the produced output of knowledge creation algorithms as knowledge, as it is information which has been adapted to its user given its local context. Both interpretations of knowledge coexist in the literature of works of knowledge creation for vehicular applications. Figure 2.5 illustrates the alternative definitions of knowledge in information systems. In both cases, algorithms or procedures able to produce contextualized content from sets of input data or information are defined as knowledge. In turn, the content produced by the knowledge procedures can be defined as knowledge or abstract information depending on the interpretation. Moreover, Figure 2.5 includes an example illustration of the differences between data, information and knowledge, through the use case of LiDAR point clouds based object recognition. In the next section, we review existing state of the art knowledge creation algorithms that have been applied to vehicular applications.

2.3.2 Knowledge Creation Algorithms

Many distinct algorithms have been defined to perform knowledge creation. Generally, computerized knowledge creation algorithms can be grouped under AI research topics. AI is a long-lasting research topic whose motivations can be broadly described as the attempt to replicate human-like intelligence in machines. Concretely, AI involves solving nontrivial and apparently complex problems, such as computer-

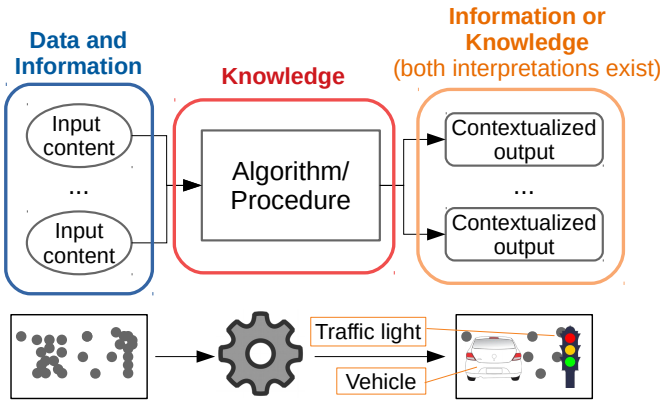


Figure 2.5: Overview of the Alternative Definitions of Knowledge

ized vision and object recognition from images, or self-driving for vehicular applications. A popular definition of AI is the study of *intelligent agents*, i.e., agents who sense and analyze their environment to take optimal decisions and achieve their set of goals [70]. To achieve their goals, intelligent agents use various algorithms to create abstract knowledge about their environment and take educated decisions. In this section, we review existing classes of state-of-the-art AI algorithms for knowledge creation which have been used in the field of vehicular applications.

Li *et al.* [71], Tong *et al.* [72], and Ma *et al.* [73] have surveyed the application of AI in vehicular networks. AI algorithms for knowledge creation in vehicular applications belong in the following categories:

- Machine Learning
- Deep Learning
- Optimization
- Knowledge-Based Systems

Machine Learning

Machine Learning (ML) and Deep Learning (DL) are the most popular knowledge creation approaches for vehicular applications, as surveyed by [72], as they represent over 50% of the share of considered key vehicular AI applications between 2009 and 2019. ML is a broad subfield of AI. It is a set of data-driven techniques which are used to learn patterns and abstract understandings as well as models from large sets of input information. As described by Ye *et al.* [74] as well as Tang *et al.* [75], who surveyed ML techniques used in vehicular applications, ML can be separated into three classes, namely, (i) supervised learning, (ii) unsupervised learning, and (iii) reinforcement learning.

Supervised Learning (SL) is a set of techniques applied to classification or regression of sensed information. SL takes as input a *training* set of information items, each associated with a class label – for classification – or a numerical value – for

regression. After a training process where the SL algorithm processes the training input, the obtained model is used to predict the label of unseen information items. To evaluate the accuracy of the trained model, a fraction of the training set is typically separated to compare the true labels with the predicted labels from the model. In turn, the obtained SL model can be considered as knowledge, as it is able to discover hidden patterns in the data and apply them to analyze unseen input information. Many algorithms have been implemented to train SL models. Solutions include linear and logistic regression, decision trees, neural networks, support vector machines, or naive Bayes classifiers.

Unsupervised Learning (UL) is a ML technique which applies to unlabeled datasets. Instead of training to recognize any underlying relationship between input data and provided labels, UL algorithms directly extract patterns and relationships within entries of the dataset. UL is used to extract clusters of data points from a dataset. The ability to group data points together according to a distance metric can be assimilated with the extraction of knowledge from the raw information contained in the dataset. Popular algorithms for UL include K-Means, K-Nearest neighbors, or hierarchical clustering.

Reinforcement Learning (RL) considers an agent interacting with an environment, taking a set of actions which are rewarded or penalized depending on the current state of the environment and the taken action. RL algorithms aim to learn a *policy* to choose the action that maximizes the received reward, based on the current state of the environment. For example, [55] and [76] implemented RL to learn how to organize traffic lights to, respectively, reduce congestion and the cumulative waiting time of vehicles. Formally, RL is defined by:

- A set of possible actions A.
- A set of possible environment states S.
- A state transition function $P : S \times S \times A \to [0, 1]$, giving the probability for the environment to transition from a state $s_1 \in S$ to a state $s_2 \in S$ when an action $a \in A$ is taken.
- A reward function $R : S \times S \times A \to \mathbb{R}$, giving the reward associated with transitioning from a state $s_1 \in S$ to a state $s_2 \in S$ by taking action $a \in A$.

In turn, the goal of RL algorithms is to learn a policy $\pi : S \times A \to [0, 1]$ which outputs the probability of taking an action $a \in A$ when the environment is in a state $s \in S$, in a way which maximizes the expected cumulative reward. As such, a learned RL policy can be assimilated to knowledge as defined in Section 2.3.1.

Model-based RL algorithms use a model of the environment, i.e, the P and R functions, to predict next states and rewards, and optimize a policy accordingly. In most cases, the analytical definition of the P and R functions is unknown to the system and RL algorithm. Instead, the agent can simply observe the reward and new environmental state obtained after taking a given action. In that case, model-based RL algorithms must first estimate a (P, R) model of the environment from experience. Model-based RL algorithms which learn a model of the environment

include I2A or MBVE, AlphaZero is an example of RL algorithm which is provided with a complete model of the environment.

Alternatively, *model-free* RL algorithms directly learn a policy π without using or estimating a (P, R) model of the environment. Rather, π is directly learned from experience and feedback from the environment after taking actions. For example, Q-learning is a popular state-of-the-art model-free RL algorithm, which has applications in vehicular networks, e.g., in [76].

Deep Learning

Deep Learning (DL) is a category of ML which specifically studies and implements complex instances of Artificial Neural Networks (ANNs). ANNs are a widespread ML technique, conceptually inspired by biological neural networks. As illustrated by Figure 2.6, ANNs are composed of a set of *neurons*. A neuron is a conceptual unit which receives multiple numerical input values and produces a single output value. Each input value of a neuron is associated with an individual weight. In turn, the output of a neuron is equal to the weighted sum of its input processed through a chosen function, called *activation function*. In an ANN, neurons are organized into sequential layers containing one or more neurons. The first layer is called input layer. Each neuron of the input layer is directly provided with user-produced input, e.g., data extracted from a training set. In turn, the output of any neuron of the layer i may be connected to the inputs of one or multiple neurons of the layer $i+1$. Finally, the output produced by the set of neurons of the last layer i_{out} – called the output layer – is considered as the output of the ANN. Intermediate layers between the input and output layers are referred to as hidden layers.

ANNs can be trained based on labeled datasets to perform supervised learning, the goal being to accurately predict the label of unseen entries. Let us consider a labeled dataset containing entries associating sets of n input features $(f_1, ..., f_n) \in F$ with sets of m class labels $(l_1, ..., l_m) \in L$. An ANN with an input layer of n neurons, any amount of hidden layers, and an output layer of m neurons can be trained so that feeding an entry $(f_1, ..., f_n) \in F$ of input features to the neuron attempts to predict their associated label $(l_1, ..., l_m) \in L$.

In order to train an ANN to produce accurate labels as output from input entries, it is trained by updating the weights of each neuron using a *back propagation* algorithm fed with real associations between training entries and labels extracted from the training dataset. The aim of ANN training is to minimize the difference between predicted and true labels, encoded in a *cost function*, e.g., the Mean Square Error (MSE) between predicted and true labels. Once trained, the ANN can be used to predict the label of unseen data by passing input entries to neurons in the input layer.

DL is a subset of ML which studies complex neural networks of multiple hidden layers. The use of multiple hidden layers combined with extensive amounts of training information allows neural networks to run several levels of abstractions of the provided input data, which improves classification performance in complex and data intensive tasks. For example, the recognition of objects such as vehicles

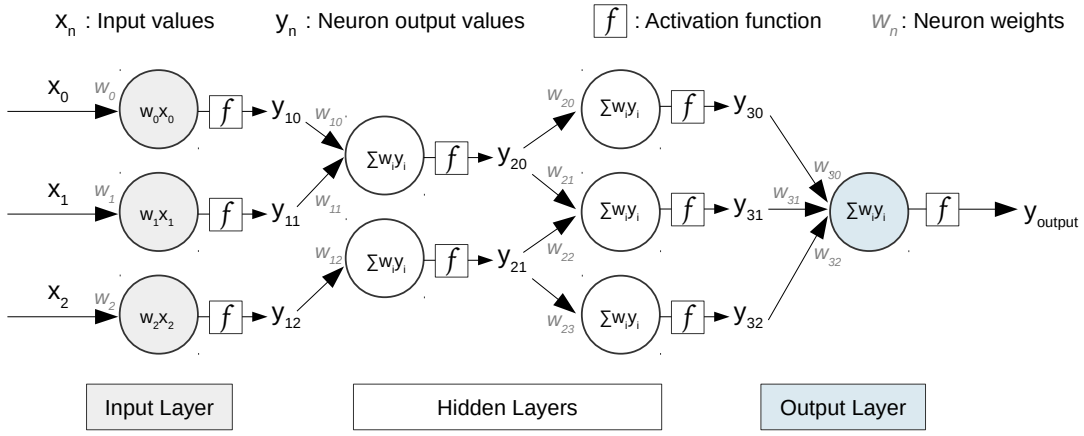


Figure 2.6: Structure of an Artificial Neural Network

or pedestrians from the set of pixels of a high-resolution camera image is a typical use case of DL. Through multiple hidden layers, the raw pixel data of the image is transformed to abstract patterns. Using extensive training input, the ANN is able to associate abstract patterns which are specific to vehicles, or pedestrians, to finally classify the object.

Optimization

Optimization is a generic term referring to a problem of applied mathematics. Based on a set of alternative elements, optimization algorithms aim at selecting the best element which fulfills a specific criterion. For example, a common use case of optimization is the selection of a local or global minimal value of a real function. Typically, optimization algorithms numerically approach a solution of the problem, i.e., finding a local minimum, or solving other criteria.

On the one hand, optimization algorithms are widely used as a support of ML and DL training algorithms, as surveyed in [77]. They aim at minimizing a *cost function* which represents the prediction error of a model being trained, e.g., the MSE between true labels and predictions for supervised learning. A popular class of optimization algorithm for ML applications is *Gradient Descent* (GD) algorithms. GD algorithms are iterative optimization algorithms which approach a local minimum of a differentiable function by iteratively moving against its gradient. GD can be formally defined using the following parameters:

- A differentiable function $F : \mathbb{R}^n \to \mathbb{R}$, $n > 0$.
- A point in the neighborhood of the local minimum $X_0 \in \mathbb{R}^n$.
- A step size factor $\mu \in \mathbb{R}^+$.

Then, we define the $(X_i)_{i>0}$ series such that $X_i = X_{i-1} - \mu \cdot \nabla F(X_{i-1})$. Provided with a 'small enough' value of μ, the $(X_i)_{i>0}$ series converges to a local minimum of F in the neighborhood of X_0.

On the other hand, optimization techniques are also leveraged directly to create knowledge as the optimal solution to solve a given problem, e.g., finding vehicle routes which minimize the overall traveling time or overall fuel emissions, as surveyed in [78]. As such, dynamic optimization algorithms themselves can be assimilated to knowledge creation algorithms.

In turn, the study of dynamic optimization algorithms to directly solve complex problems constitutes another branch of AI, called *Meta-Heuristics* (MH). As surveyed in [79], MH can be divided into (i) neighborhood-based algorithms and (ii) population-based algorithms, which include Evolutionary Computation (EvoC) and Swarm Intelligence (SI) approaches. EvoC and SI are nature-inspired dynamic optimization approaches. EvoC is inspired from the theory of evolution and includes differential evolution and genetic algorithms. SI approaches include ant colony optimization and particle swarm optimization, as surveyed in [80]. While meta-heuristics and ML are originally distinct branches of AI, modern approaches integrate ML into dynamic optimization algorithms to improve their performance, as surveyed in [81].

Knowledge-Based Systems

In the previous sections, we reviewed data-driven AI approaches. On the one hand, ML and DL are able to produce knowledge by training models to learn and recognize hidden patterns from large sets of input information. On the other hand, dynamic optimization algorithms can be directly used to produce the knowledge of the optimal solution to complex problems and have been studied as part of MH.

Knowledge-Based Systems (KBS) are an alternative to data-driven AI techniques, such as ML and DL. Rather than learning patterns from extensive sets of raw information, KBS infers output facts from sets of well-defined rules and input observations. Unlike data-driven systems, where the parameters of models trained using technologies such as ANN are not readable to human users, KBS follow explainable rules and logic inference patterns.

A KBS is composed of (i) a Knowledge Base (KB) to store a list of rules provided by human experts, and (ii) an inference engine to produce facts from input observations and rules in the KB. To express the rules of the KB, as well as input observations to provide the inference engine, a KBS combines (i) an *ontology* to give unique names to objects that will be the subject of knowledge rules, with (ii) a formal language to express knowledge as relationships between elements defined by the ontology. Existing formal languages include propositional logic, first-order logic, temporal logic, or fuzzy logic.

As an example, propositional logic is described by the following *Backus–Naur* form, a representation of its grammar along with a list of operators by precedence order.

$$Sentence \rightarrow AtomicSentence \mid ComplexSentence \tag{2.1}$$

$$AtomicSentence \rightarrow True \mid False \mid P \mid Q \mid R \mid ... \tag{2.2}$$

$$\begin{aligned} ComplexSentence \rightarrow\ & (\ Sentence\ ) \mid [\ Sentence\ ] \\ & \mid \neg\ Sentence \\ & \mid Sentence\ \wedge\ Sentence \\ & \mid Sentence\ \vee\ Sentence \\ & \mid Sentence\ \Rightarrow\ Sentence \\ & \mid Sentence\ \Leftrightarrow\ Sentence \end{aligned} \tag{2.3}$$

$$\text{OPERATOR PRECEDENCE} : \neg\ ,\ \wedge\ ,\ \vee\ ,\ \Rightarrow\ ,\ \Leftrightarrow \tag{2.4}$$

P, Q, R, ... are atomic symbols that may be `True` or `False`. Along with an ontology, the logic can describe a relationship between concepts. Let us consider an example set of sentences written in propositional logic representing knowledge with the following set of ontological symbols to bring meaning : `ICE_ON_ROAD`, `DRIVE_HIGH_SPEED`, `DANGER`, and `WINTER`.

1. `ICE_ON_ROAD` $\wedge$ `DRIVE_HIGH_SPEED` $\Rightarrow$ `DANGER`
2. `WINTER` $\Rightarrow$ `ICE_ON_ROAD`

In turn, queries can be addressed to the inference engine, e.g., `WINTER` $\Rightarrow$ `DANGER` returns *False*, whereas `WINTER` $\wedge$ `DRIVE_HIGH_SPEED` $\Rightarrow$ `DANGER` returns *True*. Yet, it is complex to model a realistic environment using only binary rules and facts which are either true or false. Instead, intuitive environmental modeling as performed by humans includes nuances and degrees of truth which vary depending on the context. As a consequence, modern applications of knowledge-based systems rely on a more complex set of logic rules referred to as Fuzzy Logic (FuzzL).

As surveyed in [82], FuzzL is a logic which integrates uncertainty to the logical rules and the inference process. Rather than being binary true of false, facts expressed using fuzzy logic have a continuous 'truth value' which can take real values in the $[0, 1]$ interval. A fuzzy expert system is composed of the following elements:

- A set of fuzzy variables organized in partitions of fuzzy sets. A fuzzy set $A = (U, \mu_A)$ is a set whose elements $u \in U$ have a degree of membership in the [0,1] interval, 1 being a full member and 0 not a member. Thus, a fuzzy set includes a *membership function* $\mu_A : U \rightarrow [0, 1]$ which associates each element of the fuzzy set with its degree of membership.

  For example, in a speed control problem, the SPEED fuzzy variable could include the `SLOW` $= (U_{slow}, \mu_{slow} : [0, 130] \rightarrow [0, 1])$ and FAST $= (U_{fast}, \mu_{fast} : [0, 130] \rightarrow [0, 1])$ fuzzy sets, which both associate speed values between 0 and 130kph with membership degrees or 'truth values' in their respective fuzzy set. For example, $\mu_{fast}(100) = 0.8$ and $\mu_{slow}(100) = 0.1$.

- A set of if-then rules of the form:

  - If `ROAD` is `SLIPPERY` then `SPEED` is `SLOW`
  - If `TRAFFIC` is `FLUID` then `SPEED` is `FAST`

- A *fuzzification* algorithm which converts real inputs, e.g., traffic density and road grip measurements, to fuzzy truth values, using *membership functions*. The membership function associated with a fuzzy set or variable, e.g., `FLUID`, converts real input values, e.g., traffic density, to truth values for the `FLUID` fuzzy variable.

  In turn, an example set of 'fuzzified' observations is:

  - `ROAD` is `SLIPPERY`, truth value: 0.4
  - `TRAFFIC` is `FLUID`, truth value: 0.7

- A fuzzy inference system to generate updated output fuzzy *membership functions* from input facts and rules. One output membership function is produced for each rule, e.g., using the 'min-max' inference method. For example, in our speed control example, updated membership functions are, respectively, produced for the `SLOW` and `FAST` fuzzy sets using the first and second rules:

  - Rule 1 $\Rightarrow$ `SPEED` is `SLOW`, membership function: $\mu_1 : [0, 130] \rightarrow [0, 1]$
  - Rule 2 $\Rightarrow$ `SPEED` is `FAST`, membership function: $\mu_2 : [0, 130] \rightarrow [0, 1]$

- Finally, a *defuzzification* algorithm takes as input the generated set of output membership functions to produce a numerical value to provide the user. For example, the obtained membership functions (μ_1, μ_2) for the partition of fuzzy sets `SPEED = [SLOW, FAST]` are used to produce a real kph speed value to be applied to the vehicle, e.g., using centroid defuzzification.

While ML and knowledge-based systems were originally two distinct and separate approaches of AI, state-of-the-art applications have combined fuzzy logic and data-driven algorithms in a novel AI technique called *neuro-fuzzy* systems. As surveyed in [83], neuro-fuzzy systems is a broad term which encapsulates various approaches combining fuzzy systems with data-driven techniques. AI algorithms such as ML, DL, or MH dynamic optimization techniques can be used to learn and optimize *membership functions* to more accurately map real world inputs with associated fuzzy truth values. Alternatively, fuzzy rules systems have been used to constraint and choose learning parameters to data-driven algorithms, e.g., DL, which improve the convergence speed of models.

2.4 Description of Knowledge

Section 2.3 presented an overview of the existing algorithms and systems for knowledge building, notably applied to vehicular scenarios, as exemplified in Table 2.1.

2.4.1 Motivation for Knowledge Description

To justify the need for means of description of a piece of knowledge, we make a parallel with information. In a centralized application which integrates information sensing and treatment in a centralized system, the system is the only entity which manipulates the information, and the only entity which formally understands the structure of the information it is treating, as it is integrated to its machine code. As the system is the only node to treat the information, no standard formal description of the information is required which is mutually understandable and interoperable to let other nodes understand the format of the information.

Yet, vehicular networks are highly decentralized and involve 'horizontal' V2V as well as 'vertical' V2I communications. What is more, they share the same goals and applications, e.g., the dissemination of safety warnings. As such, the information which is sensed by a vehicle must be understood by the remote vehicles which will receive it, so that each vehicle understands the nature of the safety warning and can react accordingly. In turn, the description of the format of the information which is exchanged is critical, and has been the subject of various standards, such as CAM [5], DENM [6], or CPM [7], as introduced in Section 1.1.2.

Similarly, it is critical to provide a description of knowledge in order to allow the networking of knowledge in vehicular networks. However, as described in Section 2.3 which described existing algorithms of knowledge creation, knowledge is a complex content which requires specific means of description to be mutually understood by the decentralized nodes of vehicular networks. In this section, we summarize the existing standards and approaches for the description of knowledge.

2.4.2 Knowledge Description Standards

As introduced in the Section 2.3, AI can be divided in two broad domains: Data-driven and descriptive knowledge. Data-driven knowledge involves ML and DL, which are able to extract generic patterns from extensive input data, and MH dynamic optimization techniques, which provide numerical solutions to complex selection problems. It is efficient to abstract knowledge from raw data, but lacks explainability. The reasons why a prediction was made, or a solution found, are mostly not human-understandable, at the exception of some cases such as decision trees for ML. On the other hand, knowledge-based systems reason on logical and human-readable rules describing the environment. However, they are not efficient enough nor adapted to modern data-intensive applications such as object recognition from raw images. While works have investigated the combination of both approaches, e.g., data-driven knowledge generating formal rules or formal rules to constrain ML training, they only apply to a reduced subset of modern applications.

Yet, knowledge is a highly contextual content as described in Section 2.3.1. In order to scale knowledge distribution and be able to apply an unknown piece of knowledge in the right context, means of description of knowledge are required. Requirements for knowledge description include the description of the interface of the knowledge, as well as of its context of training or application. If not explainabil-

(Albert Einstein, **BornIn**, German Empire)
(Albert Einstein, **SonOf**, Hermann Einstein)
(Albert Einstein, **GraduateFrom**, University of Zurich)
(Albert Einstein, **WinnerOf**, Nobel Prize in Physics)
(Albert Einstein, **ExpertIn**, Physics)
(Nobel Prize in Physics, **AwardIn**, Physics)
(The theory of relativity, **TheoryOf**, Physics)
(Albert Einstein, **SupervisedBy**, Alfred Kleiner)
(Alfred Kleiner, **ProfessorOf**, University of Zurich)
(The theory of relativity, **ProposedBy**, Albert Einstein)
(Hans Albert Einstein, **SonOf**, Albert Einstein)

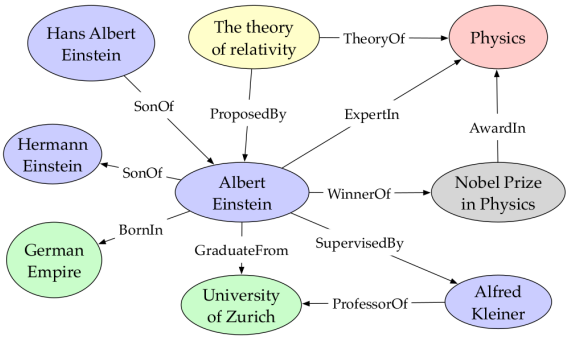


Figure 2.7: Example of a Knowledge Base and Knowledge Graph [84]

ity, the description of knowledge, including its data-driven counterpart, is a key to ensure interoperable and scalable knowledge distribution. In turn, standards have been defined to combine the performance of data-driven AI with the descriptive capabilities of knowledge-based systems.

Knowledge Graphs: Resource Description Framework

Knowledge Graphs (KG) are a form of knowledge representation. They allow the generic description of relationships between the entities and objects of an ontology. As described in [84], KG are represented by a Knowledge Base (KB) of factual triples of the form $(object_1, predicate, object_2)$, e.g., (*Right-side driving*, *AppliedIn*, *USA*). Figure 2.7 illustrates an example KB and its associated KG extracted from [84].

KG have been standardized and can be described using the Resource Description Framework (RDF) maintained by the World Wide Web Consortium (W3C). RDF aims at describing relationships between objects using freely named objects and predicates, e.g., (*Right-side driving*, **AppliedIn**, *USA*) or (*Engine*, **IsPartOf**, *Vehicle*).

Moreover, reserved predicate names are defined as part of RDF. They allow the description of relationships between different classes of objects and properties. In turn, they allow the definition of knowledge graph inference mechanisms. We provide an overview of the reserved predicates of the RDF specification. The full standard is available in [85].

type States that an object belongs to a given class, e.g., (Truck, *type*, Vehicle).

subClassOf Defines parent relationships between classes, e.g., (Vehicle, *subClassOf*, Machine).

In turn, (MyVehicle, *type*, Vehicle) implies (MyVehicle, *type*, Machine).

subPropertyOf Defines parent relationships between properties, e.g., (hasCopilot, *subPropertyOf*, hasPassenger).

In turn, (MyVehicle, *hasCopilot*, MyFriend) implies (MyVehicle, *hasPassenger*, MyFriend).

range Specifies the class of objects a property applies to, e.g., (hasCopilot, *range*, Person).

domain Specifices the class of objects a property describes, e.g., (hasCopilot, *domain*, Machine).

label Provides a human-readable name for an object.

comment Provides a human-readable description for an object.

Extensive research has been performed on the topic of knowledge graphs, and how to automate the process of learning new relationships between entities. As surveyed in [84], research is focused on two approaches: Knowledge Representation Learning (KRL) and Knowledge Acquisition (KA) aim at generating new knowledge triples to integrate in a KG. KRL learns the latent features, i.e, low dimensional representations, of objects and relationships in a KG, whose similarity is scored to learn new semantic relationships. KA notably focuses on extracting new knowledge triples by processing raw input data, with the help of AI techniques as described in Section 2.3.2.

Ontologies: Web Ontology Language

In the field of knowledge representation, ontologies formally define objects and relationships between objects which are part of a considered environment. For example, the Vehicle Signal Specification Ontology (VSSo) describes objects in the vehicular domain, such as wheels, engine, windows, speed, or cabin temperature [86].

Whereas knowledge graphs and RDF describe an environment using (subject, predicate, object) triples, ontologies build on this description by allowing the description of complex relationships between classes, objects, and properties. As an example, while RDF allows the description of any triple, ontologies acts as a 'meta'-description language which can define rules allowing a computerized system to verify if a RDF triple is valid according to ontological descriptions. Moreover, ontology description languages allow facts and new rules to be obtained from a set of facts contained in a knowledge graph through inference, using ontological relationships.

In W3C's web semantics stack, the Web Ontology Language (OWL) is the standard language to describe an ontology, to complete a RDF description. OWL follows the same triples structure than RDF. Unlike RDF however, it allows to specify complex semantic relationships between objects and classes. We provide an overview of OWL core features, i.e, the expression of (i) class relationships, (ii) individual objects relationships, (iii) property relationships, and (iv) class/property constraints. The full specification of the OWL standard is available in [87].

Class relationships Besides parent class relationships defined as part of RDF, OWL allows the expression of set-like operations between classes. Namely, union, intersection, equivalence, complementarity, and disjonction of two classes can be described through respectively, the `unionOf`, `intersectionOf`, `equivalentClass`, `complementOf` and `disjointWith` predicates.

Individual Objects relationships The OWL syntax allows declaring if two individual objects refer to the same thing, or are different. The corresponding predicates are `sameAs`, `differentFrom`, and `AllDifferent`.

Property relationships Property relationships may be described through different operation domains. Relations between two properties can be described by the `equivalentProperty` and `inverseOf` predicates. Moreover, the number of objects linked to a property can be limited using the `FunctionalProperty` – only one object is to be targeted by the property for each subject – and `InverseFunctionalProperty` – only one subject is to be associated with each object of the property – predicates. Lastly, properties can be defined as transitive or symmetric using the `TransitiveProperty` and `SymmetricProperty` predicates.

Class/Property constraints Constraints between properties and classes can be expressed with the OWL syntax. To begin with, classes can be defined based on objects matching certain properties. The `allValuesFrom` predicate and `someValuesFrom` are respectively analogous to the universal $\forall$ and existential $\exists$ quantifers of predicate logic. `allValuesFrom` can be used to define classes where all instances of a property match a target class, e.g., (hasEngine, *allValuesFrom*, ElectricalEngine) creates a class grouping all fully electrical vehicles. On the other hand, (hasEngine, *someValuesFrom*, ElectricalEngine) creates a class grouping all vehicles which feature at least one electrical engine, i.e., including hybrid vehicles. What is more, the `hasValue` predicate creates a class grouping all objects where the considered property has a given value.

Lastly, we consider the creation of classes based on cardinality constraints regarding some properties, using the `maxCardinality`, `minCardinality`, and `cardinality` predicates. For example, a class of high-capacity vehicles could be defined by (hasSeat, *minCardinality*, 20).

Knowledge Base Queries: SPARQL

The RDF and OWL description languages allow an expressive description of knowledge bases as generic relationships between objects, properties, and classes. In turn, a key requirement to use the provided knowledge description is the ability to query the knowledge base. For example, in a knowledge discovery application, vehicles and nodes of the vehicular network may wish to query for all the knowledge models which provide a given output, or all knowledge models which function with a given set of input, or in a given context.

As such, semantic querying languages to query knowledge bases have been defined as part of the semantic web stack of standards. SPARQL is a query language for RDF knowledge descriptions specified in [88]. SPARQL supports the following query modes:

SELECT Provided a list of desired output values defined by a set of conditions, a table of matching RDF objects is returned. For example, the following query will return a list of known drivers with the brand of the vehicles they drive:

```
SELECT ?brand ?driver
WHERE { ?x drives ?y ; hasName ?driver . ?y hasBrand ?brand . }
```

CONSTRUCT Instead of returning the results of a query under the form of a table of values, the *construct* operation generates a sub- knowledge graph from the input RDF data.

ASK Returns the result of a query as a *true* or *false* statement. Namely, the information that is returned is whether the query has a solution in the RDF data. For example, the following query will return *true* if at least one person in the set has at least one vehicle:

```
ASK { ?x drives ?y }
```

DESCRIBE Lastly, `DESCRIBE` is an information feature which generates a RDF-formed output which 'describes' the data which is targeted by the query. The exact format of the output and how the content is described is left at the discretion of SPARQL implementations.

The SPARQL specification also includes further querying and result formatting mechanisms, such as aggregates, and filtering of results based on functions, e.g., *IF*, *EXISTS*, logical *or* and *and*. Moreover, result presentation and ordering mechanisms are also provided. Queries are answered using a SPARQL *entailment regime*, as specified in [89]. Regimes are provided to answer queries based on standard RDF knowledge description files. What is more, SPARQL supports OWL-based entailment, which allows the resolution of queries in RDF documents completed by ontological constraints defined by OWL data. SPARQL with OWL-based entailment is referred to as SPARQL-OWL.

Services Description: OWL-S & WoT Thing Description

The RDF specification allows the description of relationships between objects in the form of triples. In turn, OWL extends RDF to provide ontologies and finer descriptions of the relationships between objects, properties, and classes. Lastly, SPARQL allows querying RDF/OWL knowledge bases to infer specific facts. As such, the web semantics stack is a state-of-the-art technology to describe knowledge and can be applied to the use case of *Service Description*.

Service Description refers to the ability to describe the interface and conditions/-context of application of a given service. Knowledge creation algorithms as defined in Section 2.3.1 can be assimilated to a service, in that they have (i) input and output interfaces, (ii) a core service intelligence, which produces outputs from input entries, and (iii) a context of application. To accurately apply data-driven knowledge models in the right context, means of description of the interface of a piece of knowledge as well as formal description of its conditions of application should be defined.

In turn, specialized ontologies have been defined as part of the W3C semantic web stack to introduce standard means of describing services, e.g., inputs, outputs,

or preconditions of application. OWL-S is a W3C standardized service description language built on top of OWL, whose specifications are described in [90]. OWL-S provides classes and properties which are adapted to the description of services and their interface.

Alternatively, recently, novel service description standards have emerged as part of the W3C Web of Things (WoT) working group. WoT is inspired from the Internet of Things (IoT) paradigm, where large amounts of physical and virtual entities are provided with network access, and produce information and services to other nodes in the network. Parallelly, in WoT, web services are the considered entities which are envisioned to be interconnected. To provide interoperability and truly produce a 'web of things', web services should in turn be semantically described so that their interface and context of application can be mutually understood.

The W3C WoT Thing Description (TD) standard provides such semantics and is fully specified in [91]. Unlike OWL-S which only provides a description of the interface of a service but does not validate the implementation of the corresponding services, TD are part of a WoT stack which includes protocol specifications to access services by standard means. To describe services, TD provides the following generic concepts:

id, title, description A unique service identifier, as well as a human-readable title and description of the *thing* and the interfaces it provides.

properties Describes the *properties*, i.e., attributes or accessible variables provided by a service. In turn, each property can be described by (i) a primitive type, e.g., integer or boolean, (ii) a unit for the quantity, (iii) a range of possible values for numbers, (iv) whether the property is read-only, (v) a human-readable title and description, and (vi) a list of *links* to access and potentially change the property value.

actions Describes the *actions* of a service, i.e., functions that cannot be achieved on a device through the modification of a property alone. Examples where actions are required include functions which simultaneously and synchronously change the value of several properties for a specific duration. Compared to properties, actions include an *input* field, which can contain a list of input parameters, individually defined as properties.

events Describes the *events* of a service. Events might be triggered by the service at any time. It is possible to subscribe to receive a notification from a service when an event has been triggered.

links Describes the *links* of a service, i.e., accessible addresses to run the properties, actions, and events of a service. Links to other resources can also be listed, such as legacy web pages.

WoT semantics and TD allow the description of the interfaces of a given service, as well as of the nature of the input it is expecting. In turn, TD has a strong potential in terms of knowledge creation interoperability, and description of data-driven AI services through semantics. As such, *Hypermedia Multi-Agent Systems* (HyperMAS)

is a recent field of research which aims at combining semantic knowledge description as provided by semantic web and WoT technologies with *multi-agent systems*.

Multi-agent systems are defined as systems composed of heterogeneous agents, each featuring distinct services which can be accessed through various interfaces. Because of this heterogeneity in agents, it is challenging to discover the abilities of each available agent. A multi-agent system can be used to request one node to conduct a task on behalf of a requesting node. As this strongly depends on how the remote node understands the task to be performed and whether the node is able to perform the task given its available services and data, WoT semantics are employed.

As in [92], HyperMAS can be used to discover and converge to a common understanding for a distributed objective between agents, and orchestrate agents to work together for specific goals, such as, e.g., model training or knowledge composition.

2.4.3 Impact on Knowledge Storage and Dissemination

The description of knowledge is more complex than the definition of a structured format for information. On the one hand, information can be described using a set of named items which can take specific values, such as in CAM [5] or DENM [6] messages. The description of information is designed to let vehicles extract specific information items. On the other hand, the description of knowledge requires the rich expression of the context of application which must be matched to use the knowledge, as well as the interface of input/output items.

Knowledge Storage

On the one hand, information databases on board vehicles are typically designed for fast information retrieval, to match the fast rate of sensing of the information. For example, the ETSI-standardized CAM messages are produced several times per second by each connected vehicle to describe their kinematics. What is more, such information has a fast obsolescence rate, and as such must be stored and accessed almost in real time. For example, the position information in a CAM which was received a few seconds ago is irrelevant as the vehicle is likely to have moved since the production of the information.

On the other hand, knowledge is a model which can treat information to produce abstract content. As such, the model itself which analyzes the information has a much longer lifetime than information, as the information processing which knowledge contains remains valid as long as it is applied in the right context. What is more, as knowledge is a highly contextual content, rich queries are necessary for vehicles to discover the right knowledge for the right context. In turn, rather than databases optimized for fast storage and retrieval of extremely fresh information, knowledge requires more organized databases which support complex querying mechanisms, such as SPARQL. Time efficiency is not as critical as with information, as knowledge can be requested beforehand by vehicles, to be used when it is needed. In turn, knowledge bases which are optimized to match rich expressive

queries, such as relational-type databases, are a potential change to implement in vehicular networks for knowledge storage.

Knowledge Dissemination & Caching

As described in the previous paragraph, a key difference between knowledge and information is that information, typically for safety applications, is short-lived and requires storage and dissemination which reduces the delay to provide fresh information to other vehicles. On the other hand, knowledge models have a longer lifetime as long as they are applied in the right context.

This has implications for knowledge dissemination processes. Information routing and dissemination in vehicular networks, i.e., especially V2V horizontal dissemination, is optimized for the fast retrieval of information. On the other hand, the dissemination of knowledge must take the context of application of the knowledge into account, e.g., to cache knowledge in the areas which feature appropriate contexts. What is more, mechanisms are required for context-based requests so that vehicles can fetch knowledge through a query for a specific interface or context of application.

Specifically, as knowledge does not have a timeout like information, but rather a context of application, it might be reused multiple times by a single vehicle, as it drives through contexts which are relevant for the knowledge. As such, it is not straightforward to know when or whether to delete a piece of knowledge on-board a vehicle, unlike information which gets obsolete with time. Moreover, connected vehicles feature a limited storage space. As such, the definition of caching algorithms for knowledge is complex. Knowledge should be dynamically cached and deleted in the locations which feature the most relevant contexts. So far, the vehicular information networking and caching mechanisms are optimized for low latency transfers of lightweight content, with local impact. On the contrary, knowledge models do not have delay constraints, but are often large files, e.g., with DL models, and have a global impact. Networking and caching mechanisms must be adapted accordingly.

Section 2.5 and 2.6 respectively describe the existing techniques and approaches for content storage and dissemination in vehicular networks. Finally, in Section 2.7, we identify problems and challenges in the state of the art, which we focus on in this doctoral work.

2.5 Content Storage and Caching in Vehicular Networks

In the previous section, we have determined a definition of knowledge. Based on the definition, we investigated state-of-the-art mechanisms and standards of knowledge creation and representation in computerized systems. In this section, we review existing content storage and caching mechanisms in vehicular networks, as a prerequisite to later investigate on any potential missing blocks to integrate knowledge storage and description mechanisms in vehicular networks.

Nodes of the vehicular network may exchange diverse types of content, including but not limited to:

- Safety notifications, e.g., accidents or road condition.
- Vehicle state information and sensor measurements.
- Navigation information, e.g., maps, road or parking data.
- Information on topics such as weather or traffic flow.
- Road-related information, e.g., gas station opening times.
- Multimedia contents for user infotainment.

Due to the development of connected vehicles, the amount of content which is generated, and potentially exchanged between vehicles, is expected to reach values in the order of 100 petabytes per month by 2025, as forecasted by the Automotive Edge Computing Consortium [93]. It comes from this that content storage and caching mechanisms in vehicular networks are a key to allow content reusability and improve scalability. Increased efficiency linked to content storage and caching in vehicular networks has been demonstrated in studies including [94] and [95].

2.5.1 On-board storage standards

A first leg of content storage in vehicular networks is performed directly on-board Connected Vehicles (CV). CVs feature an On-Board Unit (OBU), able to communicate content with other nodes of vehicular networks, perform computations, and store meaningful content. A conceptual database for content storage on-board CVs, the Local Dynamic Map (LDM), has been standardized by ETSI. Its specifications can be found in [96, 97]. As illustrated by Figure 2.8, the LDM is divided into four layers to organize content of different nature. The first layer is designed to host highly static content, e.g., map data, whereas the fourth layer is dedicated to highly dynamic content, e.g., the real-time position of pedestrians and other vehicles.

The LDM functional architecture includes two components: *LDM service* and *LDM maintenance*. The service component provides read and write access to the content in the LDM to vehicular applications. The maintenance component is designed to keep the LDM up-to-date and spatially relevant. Specifically, since the storage capacity on-board CVs is limited, LDM maintenance mechanisms have been specified to discard content as it becomes irrelevant to the ego vehicle. As such, the LDM is to be configured with an area of maintenance, which can typically be a circle of a constant radius centered on the position of the ego vehicle. Then, each piece of content which is integrated in the LDM has to come with (i) a timeout value after which it will become irrelevant, and (ii) an area of relevance. The LDM Maintenance service will discard content in the LDM if (i) it has been stored for longer than its associated timeout, or (ii) its geographical area of relevance is disjoint with the area of maintenance of the LDM.

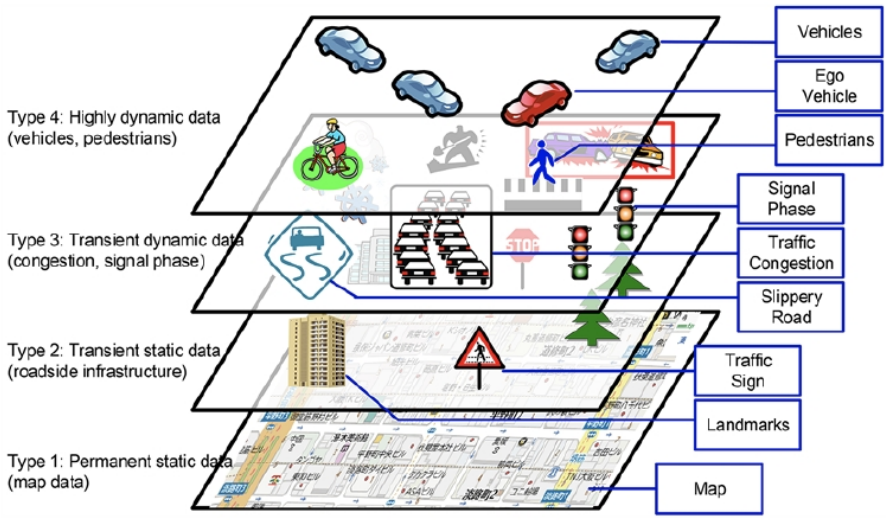


Figure 2.8: Logical Layers of Storage in Local Dynamic Maps [4]

Through this mechanism, the LDM can be kept relevant and save storage space which is limited on-board vehicles. The LDM comes with content registration and subscription mechanisms which apply to standard sets of information encoding traffic-related events, as standardized in [98]. This means that applications can subscribe to receive notifications when a message describing a standard road event is added to the LDM. Nonetheless, the LDM standard does not explicitly limit the types of content that can be stored, and implementations are free to allow any type of content in the LDM.

2.5.2 Vehicular Clouds

An approach to store vehicular data to be reused in vehicular networks is to leverage the Cloud Computing (CC) technology. In CC, the actual data storage and computation operations are offloaded to static and centralized data centers which are in turn accessed by remote nodes. A CC architecture is composed of a data center manager and a set of workers. To access or upload new content to the CC system, nodes forward a request to the data center manager, which in turns selects an appropriate worker to store or retrieve the requested content. When applied to mobile networks, the CC system is typically accessed through wireless access points, e.g., using LTE and 5G technologies. This approach has been used notably for delivery of multimedia and infotainment content to vehicles, as in [99].

The combination of CC systems and vehicular networks is referred to under the generic name of Vehicular Clouds (VC). As surveyed in [100, 101], VCs must compose with challenges that are inherent to the medium of vehicular networks, such has a high volatility and mobility of vehicles. As vehicular networks are highly dynamic and mobile environments, a constant and stable link connection to distant clouds cannot be ensured, especially in rural areas.

As surveyed by [101], state-of-the-art VCs have been realized through two broad

classes of approaches. On the one hand, a class of VCs is inspired from the architecture of conventional CC systems. We refer to them as 'conventional CC-inspired VCs'. They are formed by vehicles themselves as workers. As such, unlike in conventional cases, members of VCs are highly dynamic and may join or leave the cloud as they drive. The VC is formed by a data center manager, and a set of worker vehicles. When a vehicle joins or leaves the cloud, it issues a notification to the data center manager, which keeps a list of active members of the cloud. When an external user wishes to access the cloud, it contacts the data center manager which in turn chooses a vehicle to store the requested data, or perform the requested computation. Conventional CC-inspired VCs have traditionally been separated into static VCs and dynamic VCs. In static VCs, vehicles join the cloud when they are idle, e.g., when joining a parking lot. Dynamic VCs consider vehicles joining and leaving the cloud while moving.

On the other hand, another class of VCs interfaces vehicular networks with remote traditional CC systems. Mobile VCs are based on the assumption that vehicles do not feature enough storage and computing power on their own. In such systems, vehicles do not directly take part in a cloud architecture as workers. Instead, access points on the side of the road are used to access remote traditional clouds through wireless technologies.

What is more, it is worth mentioning *Vehicular Micro-Clouds* (VMC) as a specific case of VC. As described in [102], VMC are dynamic clouds formed by vehicles themselves, as in conventional CC-inspired VCs. What takes VMC apart is the scale of the cloud. VMCs are formed by vehicles within direct communication range, which form a virtual cloud. As VMCs are composed of low-range V2V wireless communications, they are typically formed in areas of high vehicular density, and involve vehicles joining and leaving the VMC at a high frequency.

Lastly, novel state-of-the-art VC architectures combine the technologies described in this section to form *Hierarchical VCs*. As illustrated by Figure 2.9, VMCs are formed by small and dense groups of vehicles concentrated in high-density places such as busy intersections. In turn, multiple VMCs are interconnected by vehicles in between them, which are less densely present but act as bridges between the micro-clouds. The latter are referred to as macro clouds. In turn, macro clouds are connected to edge servers and centralized data centers.

2.5.3 Vehicular Edge Computing

VCs provide the opportunity to alleviate the charge on the local on-board storage units and LDMs of CVs which are limited in storage space. Nonetheless, Mobile VCs require requests to be forwarded to a distant data center and returned to the requesting vehicle. As such, they are not suitable for time-critical applications, as described in [103]. As an alternative to Mobile VCs, works have investigated on a new approach of storing content at the edge of the network. This alternative paradigm, when applied to vehicular networks, is referred to as Vehicular Edge Computing (VEC). As surveyed in [103] and [104], VEC makes storage and computing capabilities accessible close to vehicles. In practice, Road-Side Units (RSU) or Mobile Edge

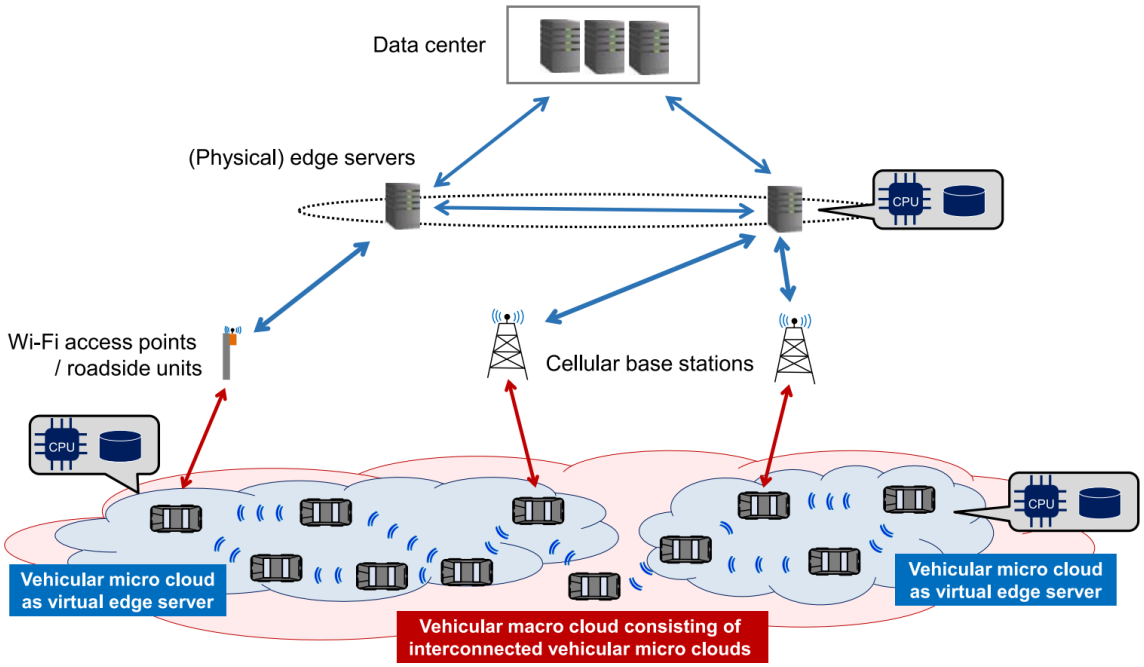


Figure 2.9: Architecture of Hierarchical VCs [102]

Computing (MEC) units are typically placed next to the road in strategic points such as busy intersections. Vehicles may issue requests for content to VEC units, such as RSUs, using wireless links. Provided the VEC unit has a cached copy of the requested content, it is directly returned to the requesting vehicle, thus drastically reducing latency compared to CC approaches where round-trip time to a distant data center must be considered.

In turn, state-of-the-art content caching algorithms for VEC units have been designed. They aim at caching content in VEC units in a way which maximizes the availability of requested content in the edge and minimizes the content retrieval latency. As reviewed in Table 2.2, various approaches have been employed to optimize content caching in vehicular edge networks. They can be classified under various categories. To begin with, vehicular mobility is a defining feature of vehicular networks which is highly predictable due to the constraints of road layouts. What is more, vehicles periodically share information about their destination, through standardized messages such as the CAM or the BSM. As such, predictions of the future position of vehicles can be used to proactively cache content at the edge of vehicular networks, as in [94, 105].

Moreover, vehicular mobility-based approaches have been combined with content popularity-based approaches. By assessing the popularity of content items, and in some cases predicting future content popularity, caching systems are able to cache the content which is most likely to be requested by vehicles, and in turn improve cache hit rates. An alternative to generic content popularity is the definition of *interest groups* among CVs. By identifying different clusters of vehicles based on their interests in terms of content type, content can be cached in a way which maximizes the cache hit ratio for popular content across various interest groups, as in [95, 106].

An alternative approach to content caching considers the problem of the wireless channel link quality between vehicles and VEC units. Because of the high mobility and dynamism in vehicular networks, as well as the imperfect coverage of VEC units, vehicles might be out of reach when they request content to the edge. In turn, approaches use caching redundancy to minimize the impact of unstable channel links between vehicles and VEC units, as in [107, 108].

Lastly, some papers have predicted the future needs of vehicles in terms of service access. By assessing which piece of content/service is more likely to be requested in the near future by vehicles, caching policies can be updated, as in [109]. All the approaches to VEC content caching have been implemented using various AI techniques, such as RL or optimization algorithms.

Table 2.2: Classification of State-of-the-Art Vehicular Edge Caching Approaches

Reference	**Edge Caching Algorithm**	**Description**
[94], [105]	Vehicular mobility-based	Extraction of vehicular mobility patterns, prediction of future vehicle positions.
[94], [110], [111]	Content popularity-based	Estimation of current and future content popularity.
[95], [106]	Interest/Social-based	Social groups of vehicles are learned based on the interests of each vehicle in terms of content type. Content is cached based on the various interest patterns of vehicles.
[107], [108]	Channel link quality-based	Content is cached in order to minimize the impact of unstable links between vehicles and VEC units.
[109]	Service prediction-based	Content is cached based on predictions of the future needs of vehicles.

2.6 Content Distribution in Vehicular Networks

Once content has been stored in vehicular networks, it has a strong potential to be distributed and re-used by vehicles and other nodes. We have touched upon the subject of vehicular networking by introducing the topics of vehicular clouds and vehicular edge computing. In this section, we review more specifically the state-of-the-art protocols and standards which are in use to support content distribution in vehicular networks. Then, provided with an understanding of the existing approaches for content storage and distribution in vehicular networks, as well as a description of knowledge and knowledge creation mechanisms, we open the possibility to analyze whether vehicular content management standards are sufficient and adapted for knowledge creation and distribution in vehicular networks.

2.6.1 Mobile Ad-Hoc Networks

Mobile Ad-Hoc Networks (MANETs) are networks which consider the ad-hoc, wireless interconnection of nodes that move in a highly dynamic topology. Figure 2.10

illustrates a typical MANET scenario, where a packet has to be routed from a source to a destination through multihop ad-hoc wireless communications. Before the advent of vehicular networking and VANETs, which are a specific case of MANETs, extensive research was conducted on architectures and routing protocols to support the multi-hop networking of content among MANETs.

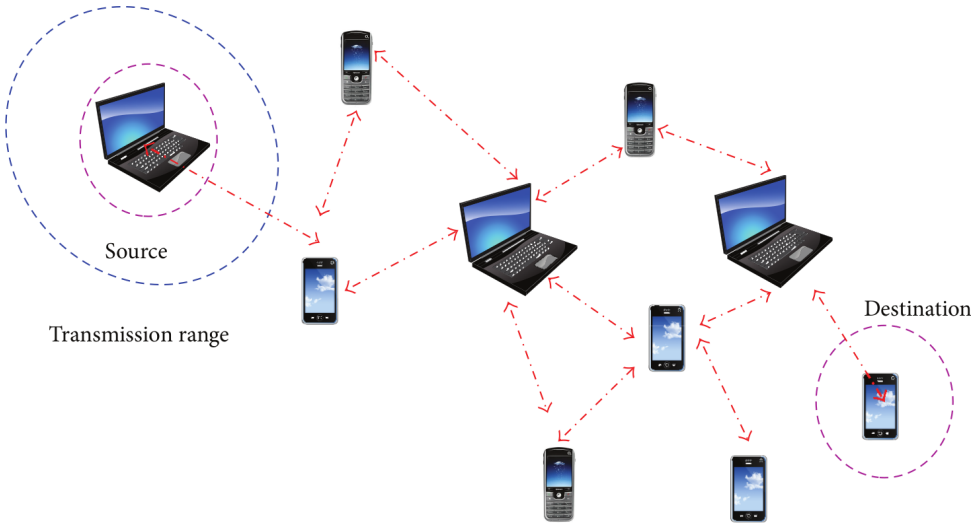


Figure 2.10: Overview of a MANET Scenario [112]

The main characteristics as well as a definition of MANETs have been published in the Internet Engineering Task Force (IETF) RFC 2501 memo [113]. Namely, MANETs involve four key aspects:

Dynamic Topologies Nodes can move arbitrarily, in different speed and directions. In the specific case of vehicular networking, nodes typically follow vehicular mobility.

Bandwidth Constraints Due to the constraints of the wireless medium, the bandwidth capacity is significantly lower than with wired connections.

Energy Constraints Nodes of MANETs typically rely on batteries with limited available power. As such, MANETs packet dissemination mechanisms should be energy-efficient.

Security The wireless medium is physically more prone than wired alternatives to security-related attacks such as eavesdropping, spoofing or denial-of-service.

In turn, at the early stages of mobile wireless ad-hoc networking, routing protocols have been defined to disseminate content in MANETs, as described in [114]. They are divided in three main categories, namely, (i) proactive, (ii) reactive or on-demand, and (iii) hybrid protocols. Proactive algorithms actively exchange route discovery packets to maintain full knowledge of the available routes in the mobile network as the topology evolves. *Optimized Link State Routing* (OLSR) [115] is an example of a proactive MANET routing protocol. On the contrary, reactive algorithms compute routes on demand, as it is needed, i.e., when a packet has to be transmitted. *Ad-hoc On-demand Distance Vector* (AODV) [116] and *Dynamic Source*

Routing (DSR) [117] are examples of reactive protocols. Finally, hybrid protocols combine reactive and proactive aspects. The *Zone Routing Protocol* (ZRP) [118] is an example of hybrid MANET routing protocol.

The routing protocols defined for MANETs aim at adapting the existing Internet protocol for the highly-constrained mobile wireless environment. In turn, the communication is host-centric, i.e., MANET nodes send or receive information by explicitly referring the address of the source and destination of the information communication. This paradigm is called CDN. This is well suited in MANET scenarios such as sensor networks, where sensors provide information to a well-known gateway host, which will in turn expose the information to the Internet through a specific interface. Namely, MANETs and VANETs did not consider how to: (1) know the destination, (2) know the source, (3) know what actually needs to be transmitted, it was purely a network optimization field.

Yet, a key specificity of VANETs, which adds to the MANET characteristics as defined in [113], is the fact that connected vehicles are both producers and consumers of information. As connected vehicles produce their own information at a high frequency, it is not trivial to know which host is in possession of which information. When looking for a piece of information, in order to use traditional MANET routing protocols such as AODV or OLSR, the host address of the vehicle which owns the wanted information must first be determined. This is challenging for two reasons. To begin with, it is nontrivial to obtain the address of a vehicle which owns the wanted information, given the highly dynamic topology and the fact that each vehicle produces its own information. What is more, in many vehicular applications, such as infotainment content retrieval, the information itself matters more than the host it was obtained from. In turn, protocols were defined to emancipate VANETs from the need to explicitly refer to a specific host to retrieve information, but instead directly request a copy of a named information. This alternative networking architecture is referred to as information-centric networking.

2.6.2 Information-Centric Networking

Traditionally, nodes and services of the Internet are interconnected following a CDN approach. Large data centers hold extensive amounts of information which can be downloaded by nodes provided the knowledge of a host address where the information can be fetched from. CDN is currently the prevalent paradigm for information distribution in use in the Internet. As an alternative to CDN, ICN is a paradigm which uncouples data from its location. Rather than by host address, information is addressed by its name. Combined with information caching and routing mechanisms, ICN allows to alleviate the network load on traditional data centers. As many vehicular applications are focused on consuming information, regardless of its host, ICN is well-suited for vehicular networking.

Conceptual Description

Information-centric networks are composed of three main building blocks: information routing, caching, and naming. ICN routing schemes should efficiently route information requests to their respective producers. Caching mechanisms are a key to reduce the number of hops required to retrieve a given piece of information. Finally, a naming scheme must be implemented to identify information items.

As surveyed by [119], routing schemes have various low-level implementations, but fall into two broad categories:

- In *pull-based* approaches, nodes issue an *interest* for an information name, which is multi-hop forwarded to content producers using various forwarding algorithms, as surveyed in [119]. Once a content producer or a node with matching cached information receives the request, the information is forwarded back to the source of the interest.

- In *publish & subscribe* approaches, information consumers declare their interest for specific subsets of content names. When content producers issue a new piece of content matching the subscribed name, it is asynchronously forwarded to each subscribing node.

To reduce the workload of content producers, as well as reduce the access time to information, named information may be cached in the nodes of an ICN network using various schemes, as surveyed in [120]. When an interest message for a named information is forwarded through a node which owns a copy of the requested information in its local cache, the information is directly returned before reaching a content producer.

As surveyed in [120], schemes for information naming in ICN implementations can be grouped into the following categories:

- *Flat* names are systems which map a piece of information with a monolithic name. For example, a piece of information about the road roughness in a specific road section r could be named `road_roughness_in_`r.

- *Hierarchical* names are organized following a hierarchy of content names from generic categories to specific information items. For example, a hierarchical name for the previously defined information item could be: `/road-information/road-`r`/roughness`.

- In *attribute-value-based* names, information is named according to a set of descriptive pairs of (attribute, value) parameters. Consequently, a single name does not necessarily refer to a single piece of information.

- Finally, *hybrid* approaches combine two or more of the previous naming systems, and may integrate algorithmic novelties, e.g, query-based names as introduced in [121].

Named Data Networking

Named Data Networking (NDN) is a popular implementation of the ICN paradigm, introduced for vehicular communications in [122]. It is a network-level protocol which handles the routing of interest packets from content consumers to content producers. It is *pull-based* and content items feature hierarchical names, i.e., content belongs to hierarchical named categories, which are in turn used for routing purposes.

As described in [122], routing of content in NDN networks is performed in two steps. First, the content consumer issues an *interest* packet which contains the unique name of the requested content, e.g., `/road-information/road-`r`/roughness`. In turn, the interest packet is routed to a content producer which is able to produce the requested named content, potentially through multi-hop forwarding. Finally, the content producer returns a *data* packet, which is routed back to the original requester through the same path, backwards.

The routing of interest and data packets in NDN networks is performed by *NDN Forwarding Engines* (NFE) which are implemented in each node of the network. NFEs are composed of the following elements:

Forwarding Information Base (FIB) The FIB is a database which contains a set of known $(prefix, next\_hop)$ pairs. Upon reception of an interest packet for a named content, the longest available $prefix$ which is a prefix of the content name is extracted. In turn, the interest packet is forwarded to the corresponding $next\_hop$ node. Various approaches are possible to discover new entries to add to the FIB, such as producer announcements or flooding.

Pending Interests Table (PIT) The PIT is a database containing a list of interest packets which have been forwarded through the NFE but for which no answer data packet has yet been received by the NFE. Elements of the PIT are of the form $(name, interest\_nonce, incoming\_hop)$, where (i) $name$ is the name of a received interest, (ii) $interest\_nonce$ a unique interest identifier, and (iii) $incoming\_hop$ the hop from which the interest packet was received. When the *data* packet matching the interest is received by the NFE, the corresponding PIT entry is removed, and the data packet is forwarded back to the associated $incoming\_hop$. The process is repeated recursively until the data packet reaches the original interest emitter.

Content Cache The content cache is a database used to store cached copies of the data packet which have been forwarded through the NFE. The content cache can reduce the number of hops associated with the routing of an interest packet. During the multi-hop process of forwarding the interest packet to an associated content producer, if an intermediate NFE contains a cached copy of the requested data packet, the interest routing is stopped and the data packet directly returned from the cached version. Then, the content is returned to the original interest emitter without having needed to route the interest all the way to a content producer. Various caching schemes can be implemented as part of NDN, such as freshness-based or popularity-based systems.

2.6.3 Function-Centric Networking

ICN and NDN have been designed to support information exchange. Nodes can issue requests for a copy of a named information, which will be forwarded back to them. Yet, this approach only permits the distribution of static copies of the information, and does not allow nodes, e.g., vehicles, to fetch personalized pieces of content, which have been preprocessed to match specific requirements. To address this challenge, an alternative paradigm built on top of ICN has been studied in the literature, which we refer to under the broad name of *Function-Centric Networking* (FCN).

Generally, FCN approaches are composed of two elements: content items and functions. As with ICN, content item are uniquely named pieces of content which can be retrieved upon request. On the other hand, functions are new items which can be manipulated through FCN. FCN functions are named functions which take as input a named content item, perform some processing, and produce an output. In addition to being able to request static copies of named content items, nodes of a FCN network may request to retrieve a content which has been produced by the remote application of a named function to a named content item. For example, let us consider an object recognition problem from drone-captured road images. Using ICN, the consumer vehicle directly requests a copy of the `/road-`r`/drone-image` content and locally processes the image with AI algorithms to perform object recognition upon reception. Using FCN, the AI function performing object recognition can be encapsulated in a FCN function, e.g., named `/function/object_recognition`. The function is made available by a set of nodes in the network. In turn, the consumer vehicle can directly request for the `/function/object_recognition ( /road-`r`/drone-image )` content, i.e., the result of the application of the object recognition function on a drone image content.

Various FCN approaches have been implemented, which differ in terms of (i) the definition, form, and implementation of functions, and (ii) the routing algorithm which routes the interest to fetch all three of the input content, the named function, and decides where the computation will occur before the result is returned to the original request emitter. Named Function Networking (NFN) was the original proposal for the concept of function-centric networking. It is described in [17]. As suggested by its name, it is an extension of NDN, which introduces the in-network execution of named functions to NDN data items. In NFN, functions are implemented as λ-expressions. Namely, nodes can request the in-network application of a chain of named functions in the form of a λ-expression to a target piece of named information. Alternative function-centric systems include NFaaS [123] and NDN-FC [124] which network functions encapsulated in virtual images or containers instead of λ-functions.

2.6.4 Knowledge-Centric Networking

Lastly, Knowledge-Centric Networking (KCN) is a conceptual new content delivery paradigm which aims at focusing on knowledge delivery rather than information delivery. ICN exchanges copies of static information. FCN goes further by allowing

the in-network treatment of static information. In turn, KCN aims at investigating on the conceptual differences between information and knowledge, and the novel types of networking mechanisms which are required to obtain the full potential of knowledge networking. The KCN concept has been introduced in [125]. It generally states that KCN involves the aspects of knowledge creation, storage and distribution. Knowledge creation includes the composition of several existing pieces of knowledge to produce new knowledge.

In parallel, applications have been defined in the literature which applied the concept of KCN, without formally being referred to as KCN. In [126], knowledge is created as a ML model predicting the video playback pattern of users. The knowledge is sent to edge servers to optimize video caching. In [127], statistical knowledge about network topology is used to optimize routing in unmanned aerial vehicle fleets.

While the concepts of KCN have been described in [125] and applied for specific applications in works such as [126, 127], the papers provide no formal structures for generic knowledge representation or protocols for knowledge distribution. Alternatively, Sapra *et al.* [128] proposed a KCN architecture where knowledge is expressed and composed as DL models. The architecture and weights of ANNs trained as part of DL applications are described and exchanged between nodes. In turn, the layers of DL models might be swapped or replaced by layers obtained from models trained in different conditions. The authors provide several case studies to demonstrate potential applications, i.e., object and activity recognition, where exchanging models weights instead of training data allows bandwidth gains of up to 60%.

2.7 Discussion

Vehicular networking has been defined as an enabler for smart applications, to improve safety, traffic efficiency and provide infotainment within networks of connected vehicles. Due to the key challenge of the vehicular environment which is the high mobility of nodes, leading to unstable channel links, specific networking protocol stacks have been defined, either based on WiFi or cellular technologies.

Furthermore, another key aspect of vehicular networks which differs from traditional MANETs is the fact that each connected vehicle both consumes and produces information, which is distributed to other nodes. What is more, the produced information, e.g., vehicle kinematics data for safety applications, is typically short-lived. This makes it challenging to know precisely which host is in possession of which information in vehicular networks. In turn, specific information-centric networking protocols were designed which uncouple information from its host, i.e., requests can be routed directly to a specific named information item.

In modern vehicular applications, vehicles are evolving from *connected* to *intelligent*, and knowledge is increasingly used in various aspects of vehicular networks, as exemplified in Table 2.1. Through AI algorithms, knowledge can be trained to provide abstract models and understandings of the environment, built from observed information. Namely, knowledge has been used to (i) optimize the networking operations themselves, (ii) optimize the caching and storage of information and computing

resources, and (iii) as part of various vehicular applications, such as traffic light of highly-automated vehicle control.

As such, vehicles not only use knowledge which was built in centralized data centers, but also cooperatively build their own knowledge, to be shared with other nodes. For example, in [46], vehicles cooperatively learn knowledge to reduce the power consumption of their communications. In [65], vehicles build a shared model to predict the range and energy demand of electric vehicles.

In turn, just like a specific set of protocols was designed to allow information networking in vehicular networks where each vehicle produces their own information, mechanisms should be defined to build *vehicular knowledge networking* where vehicles can cooperatively create, store and disseminate knowledge.

In this chapter, we reviewed the definition and algorithms to create and describe knowledge, providing an understanding of the key differences between information and knowledge. In turn, we investigate the limitations and open issues which should be addressed for vehicular knowledge networking.

2.7.1 Knowledge Description

The information which is exchanged between vehicles, e.g., for safety applications, such as the ETSI CAM [5] which provides kinematics data about a vehicle, typically has a very low time of validity. As such, mechanisms for information storage and dissemination in vehicular networks are optimized to provide low delays. This applies to (i) inserting and extracting information from the LDM vehicular database, (ii) exchanging information through specific channels or protocols for low delay delivery of safety information, or (iii) information-centric approaches which cache information close to its expected consumer.

On the other hand, rather than its freshness, the relevance of a knowledge model is measured by its accuracy in the driving context it is used in. As such, the first key aspect for knowledge storage and dissemination is to build a description of knowledge and its context of usage, as considered in Section 2.4. The semantic of AI, and specifically ML, models must be identified and described, including their input and output interface, as well as their context of use and training, to ensure that they are used in the right context. Namely, there is a need for the definition of a form of knowledge graph for AI/ML knowledge which is used in vehicular networks, which states (i) what it is, (ii) what it is used for, (iii) where it can be used, (iv) where it has been trained, and (v) how to use it, i.e., what type of input information it needs, and what type of output it produces. This description could be performed using Semantic Web or WoT technologies. Generally, semantic description standards and technologies could be used to support the distributed creation, storage, and dissemination of knowledge in vehicular networks.

Namely, through a mutually understandable semantic description of the input/output interface of a knowledge model, as well as a description of the context in which the knowledge should be used to yield accurate results, nodes of vehicular networks could be able to understand and use knowledge which was produced by

another node.

2.7.2 Impact on Knowledge Storage

So far, information bases such as the LDM on-board vehicles are optimized to insert and extract information at high speeds, matching the need for low delay in the dissemination of safety information. However, the delay of inserting and extracting knowledge models from a database is not critical, as the knowledge is valid for extended periods of time, as long as it is applied in the right context. As such, even with higher latency to access models, knowledge can be requested proactively by vehicles, before they need to use it.

Rather, knowledge storage requires a database which lets vehicles query knowledge, not based on the time it was inserted, but on its interface and context of application, to discover the right knowledge for the right vehicle and driving context. Yet, no mechanism has been specified as part of the LDM standard [96] to issue rich semantic queries about its content. What is more, the map-levels layered architecture of LDM is not directly relevant for knowledge, which is not necessarily associated with a unique location, but can be relevant for multiple contexts.

What is more, the access to LDM content is done by publish and subscribe, through standardized semantics [96]. Instead, the storage of knowledge requires a content access mechanism which supports generic, complex, and rich queries based on knowledge description semantics, which might be partly undefined. For example, a vehicle might wish to discover all the existing knowledge which produces a specific output, or which can be used in a given context, without providing the full semantic description of this knowledge. In turn, relational databases which can efficiently handle requests for knowledge with specific context conditions, e.g., answer to SPARQL queries as introduced in Section 2.4, would be adapted for knowledge.

2.7.3 Impact on Knowledge Dissemination

Similarly to knowledge storage mechanisms, knowledge dissemination mechanisms should support requests for the discovery of knowledge in a given context, and operations which are specific to knowledge, such as cooperative knowledge training or knowledge as a service, i.e., sending requests to a remote node for the application of a knowledge model it possesses.

Yet, while information is associated with a specific geographic and time scope of validity, which bounds the area and time in which it can be disseminated, knowledge is associated with a context of validity. In turn, a valid context of application of knowledge can be matched in multiple distant locations, and for extended periods of time. What is more, the locations which feature a valid context for the knowledge may evolve through time. For example, a model which recognizes and classifies objects from camera and LiDAR sensors, as in [60], and has been trained in California in urban conditions and clear weather, will be relevant in urban areas of California which feature clear weather.

As such, the duplication of AI knowledge models, and their dissemination in the contexts which are currently relevant to them is required. From a service perspective, knowledge needs to be identified and gathered. ICN and FCN networking can provide a foundation to support the distribution of knowledge discovery or knowledge creation queries, assuming the knowledge model is well described through semantics. Yet, applied to the NDN technology, the key paradigm which assumes that information is likely to be stored close to where it was created, needs to be reinvestigated. What is more, in information retrieval cases, if a copy of the information is found in a NDN cache, it is directly retrieved to the interest emitter. Yet, in the case of remote knowledge application, mechanisms are required such that a NDN cache which owns a copy of a knowledge model may not only transmit the copy of the model, but also directly apply the knowledge and return its output.

In turn, early knowledge-centric networking applications have researched the networking of knowledge in specific applications. For example, [125] described the high-level requirements for knowledge networking and identified the need for knowledge creation, storage and dissemination approaches. In turn, [126, 127] have applied knowledge networking to specific use cases, but without implementing generic means of knowledge networking based on knowledge description. Finally, [128] has provided an architecture which is heavily centered around DL models and their in-network composition to achieve models which are better adapted to a context. Yet, it does not include a generic means of describing the context of application of a generic AI knowledge, such as mathematical optimization or knowledge created from original algorithms.

In this discussion, we identified limitations in three key aspects of the current vehicular networks, namely, (i) knowledge description, (ii) knowledge storage, and (iii) knowledge dissemination. In turn, in Chapter 3, we analyze how to address these limitations for each key aspect. Based on this analysis, in Chapter 4, we define a framework for knowledge networking in vehicular networks, which addresses these three key aspects. Finally, in Chapters 5 and 6, we define a realistic knowledge model of roundabout driving risk, to evaluate its distribution through the framework.

Chapter 3

Analysis of Knowledge Networking in the Vehicular Environment

In Chapter 2, the current architecture of vehicular networking for information storage and dissemination was summarized. What is more, a definition of knowledge as well as an overview of the AI algorithms to build knowledge was provided. In turn, in Section 2.7, we identified the key differences between information and knowledge. Namely, while the relevance of information is based on its freshness, e.g., in safety applications, the relevance of knowledge depends on the context in which it is used by vehicles. For example, when applying a model to assess the probability of a vehicle to exit a roundabout, a model trained on a 7-entry right-side-driving roundabout in the USA might not be relevant for a 3-entry left-side-driving roundabout in Australia.

In turn, we identify three open challenges in the existing protocols of vehicular networking to support knowledge networking:

1. To begin with, due to the key impact of context for knowledge, it is critical to define means of description of the context in which a knowledge model should be applied.

2. In turn, knowledge databases must be adapted to optimize rich context-based queries to discover appropriate knowledge, rather than databases optimized for fast inserting and extraction of content.

3. What is more, knowledge dissemination mechanisms which integrate the context of application or the context of training of a model are critical. Given the nature of the vehicular environment where each node moves and has a highly volatile training environment and driving context, communications must be developed for vehicles to exchange about their driving environment, and the existing models which match this environment.

In this chapter, we perform a detailed analysis to stress the extent of the problematic for each point, and analyze the challenges which must be addressed to overcome them and allow vehicular knowledge networking.

In Section 2.3, we performed an overview of AI algorithms to create knowledge. Table 2.1 illustrated the wide range of applications for knowledge in vehicular networks, and the diversity of algorithms to build it. Yet, we observe that ML and its variants, e.g., reinforcement learning, deep learning, or federated learning, are especially common in vehicular applications of knowledge. As such, in this analysis, we focus on the case of ML models to exemplify knowledge description, storage and dissemination challenges. Yet, the approaches and findings we make are generic to any form of knowledge as defined in Section 2.3.

3.1 Analysis of Knowledge Description

To begin with, we provide a scale of knowledge building operations based on the level of distribution and cooperation between vehicular nodes which it implies. In turn, we analyze the criticality of knowledge description for each aspect. The contribution of the classification of the levels of distribution of knowledge in vehicular networks is given in Figure 3.1 and is referred to as the **scale of knowledge distribution** in the rest of this doctoral work.

Depending on the level of 'distribution' of the knowledge application and building operations, the definition of semantic descriptions of knowledge is critical at various levels. In Section 3.1.1, we analyze the impact and the critical need for a mutually understandable semantic description of knowledge for each case of knowledge distribution.

3.1.1 The Key Aspect of Knowledge Semantic Description

To introduce our discussion on the importance of semantic knowledge description in various levels of the **scale of knowledge distribution**, we consider the fully centralized case, i.e., **level 0**, where the knowledge is trained and used exclusively by a single node, such as for example, a traffic light optimizing its own control to improve traffic flow. In this case, no exchange of knowledge is required whatsoever, the knowledge is hard-coded in the node which uses it. In turn, the node is provided with a *de facto* description of how to use this knowledge, and no explicit interoperable semantic description is required.

Nonetheless, the level 0 of the scale of knowledge distribution is the exception rather than the rule in vehicular environments, which are typically highly distributed. In turn, we consider the need for a semantic description of knowledge in cases where knowledge is more distributed.

Level 1: Knowledge Sharing for Personal Use

Another key aspect of knowledge distribution is the **level 1** of our scale of knowledge distribution. It is common in vehicular networks, as illustrated by Table 2.1. It refers to the distribution of knowledge which has been trained in a centralized unit, e.g., a data center or a single vehicle, to other vehicles. For example, [60] trains

Level 0 The knowledge is trained by a single node, for its exclusive use.

- For example, a model trained on an infrastructure node, whose results are relevant to the same infrastructure node. In [55], knowledge is trained to control a traffic light node to reduce the waiting time of vehicles.
- Impact on the need for describing semantics: *None.*

Level 1 The knowledge is trained by a single node, and shared with other vehicles for their personal use.

- For example, in [60], a model is trained to recognize objects from camera and LiDAR images on a centralized data center. Yet, the knowledge is meant to be downloaded to vehicles, so they can use it locally. Moreover, vehicles can use the knowledge independently from their own front camera images, and no cooperation with other vehicles is required to use the knowledge.
- Impact on the need for describing semantics: *Input/output interface, context of use.*

Level 2 The knowledge is trained on a vehicle, and shared with other vehicles for distributed use.

- For example, in [47], a model is trained to recognize the patterns of transmission of other vehicles, in order to avoid collisions in the wireless MAC layer. The knowledge is trained on a single node, but is meant to be used cooperatively, as the prediction of transmission patterns in presence of more than two vehicles requires coordination among vehicles.
- Impact on the need for describing semantics: *Cooperative task description, driving context of vehicles.*

Level 3 The knowledge is directly trained cooperatively by a pool of vehicles.

- For example, in [46] and [65], vehicles cooperate to learn a model in a distributed way, respectively, to learn network queue patterns for reducing the energy consumption of low latency communications, and to predict the energy demand and range of electric vehicles.
- Impact on the need for describing semantics: *Interface of the trained model, training context, training algorithm, training algorithm parameters.*

Figure 3.1: Our Scale of Knowledge Distribution

a model to recognize objects from sensor data, [45] predicts the lifetime of direct V2V communication links for vehicles to choose stable communication routes, or [58] estimates the road surface quality from camera images, among others.

Using the Knowledge On the one hand, once a vehicle receives a copy of a newly trained knowledge model which has been produced by a remote centralized node, it does not know how to use the knowledge model. Namely, a semantic description is critical to describe the set of inputs as well as the set of outputs which are produced by a model. In turn, when receiving a copy of the object recognition model described in [60], the vehicle is aware that the input value to give the model is a front camera image and a LiDAR point cloud.

Furthermore, as the relevance of a piece of knowledge highly depends on the driving context it is used in, a description of the relevant context of use of knowledge is necessary. For example, considering that the object recognition model in [60] has been trained from data collected in a sunny environment, the model may not detect objects accurately under heavy rain or snow. As such, the elements which make a relevant context to use a specific piece of knowledge must be described, so that the knowledge is used in the right environment.

Discovering the Knowledge What is more, as vehicles move, the list of other nodes with which they can have a direct V2V or V2I communication dynamically changes all the time. In turn, there is no straightforward way for vehicles to get a global view of the knowledge which is available to them in the first place. Thus, semantic description is also critical to let vehicles discover and use the knowledge which is located in remote nodes, by requesting knowledge based on its input, output, or context of application.

For example, if a vehicle would like to obtain a model to recognize objects on a road in sunny conditions, it needs to be able to use interoperable semantics to ask other nodes whether they know about a piece of knowledge (i) which produces object classification, (ii) in a sunny weather context. All these aspects are critical and must be supported through well-defined semantics.

Current Approaches So far in ML model applications for vehicular networks, it is assumed that all nodes (i) possess full knowledge of the existence of the considered model, and (ii) are using the model in the right context. On the one hand, research papers which describe models typically use validation data which is extracted from different entries of the same dataset which was used for the training. As such, the context of application is *de facto* relevant. Yet, the distribution in real vehicular scenarios is not considered. On the other hand, models which are deployed in production by the industry in connected vehicles are trained to be distributed in a single fleet of vehicles, which belong to the same automaker. In turn, the input/output interface as well as the context of application of the model is encoded in a proprietary format on-board vehicles.

On the contrary, presented with new knowledge, what if no description semantic

is available? The model cannot be applied without knowing its input parameters, and its output is unclear if its type is not known. What is more, how to know if the model is being applied in the right context? As everything moves in vehicular networks, knowledge cannot be discovered, let alone used without mutually understandable semantics.

Level 2: Knowledge Sharing for Cooperative Use

While knowledge which has been fully trained by a centralized node can be distributed to vehicular nodes for local use, such as with the object recognition model in [60], some knowledge models require cooperation between nodes to be used.

For example, [47] trains a model to learn the patterns of transmission of other vehicles in order to infer when to send packets. However, in complex cases where more than two vehicles are present, the authors note that the vehicles must cooperate to learn an optimal transmit pattern.

While they are not ML algorithms, we can quote the following works which have defined knowledge which is meant to be used cooperatively and in synchronization with other vehicles: [129] described knowledge which requires active synchronization among vehicles crossing an intersection to increase safety and reduce the likelihood of accidents. Similarly, [130] defined a model to be used simultaneously by several vehicles to cooperatively plan maneuvers such as overtakings, and [131] provided a cooperatively used knowledge for the dynamic control of platooning, i.e., self-organizing queues of vehicles in a highway.

Furthermore, Multi-Agent Decision Making (MADM) is defined as a process where multiple entities provide input knowledge to take global decisions aiming at achieving common objectives, as surveyed in [132]. As such, MADM systems require the cooperative application by several vehicles of a trained knowledge in order to reach a consensus on a decision to make, in various use cases.

As with *level 1* models, which are meant to be used locally, without synchronization with other nodes, a semantic description must describe the input, output, and context of application of the knowledge, e.g., the type of intersections it applies to. What is more, specific semantic description mechanisms are required for vehicles to provide or request information about their current driving context, such that vehicles which feature the right context and possess the same piece of knowledge can form groups to cooperatively use the knowledge.

Namely, if no such semantic is defined, due to the highly dynamic topology and fast evolving driving context of each vehicle on the road, potential other nodes which could cooperate to apply a given piece of knowledge cannot be detected and groups cannot be formed, as it is unknown whether neighboring vehicles possess the right knowledge, and the right driving context.

Level 3: Knowledge Cooperative Training

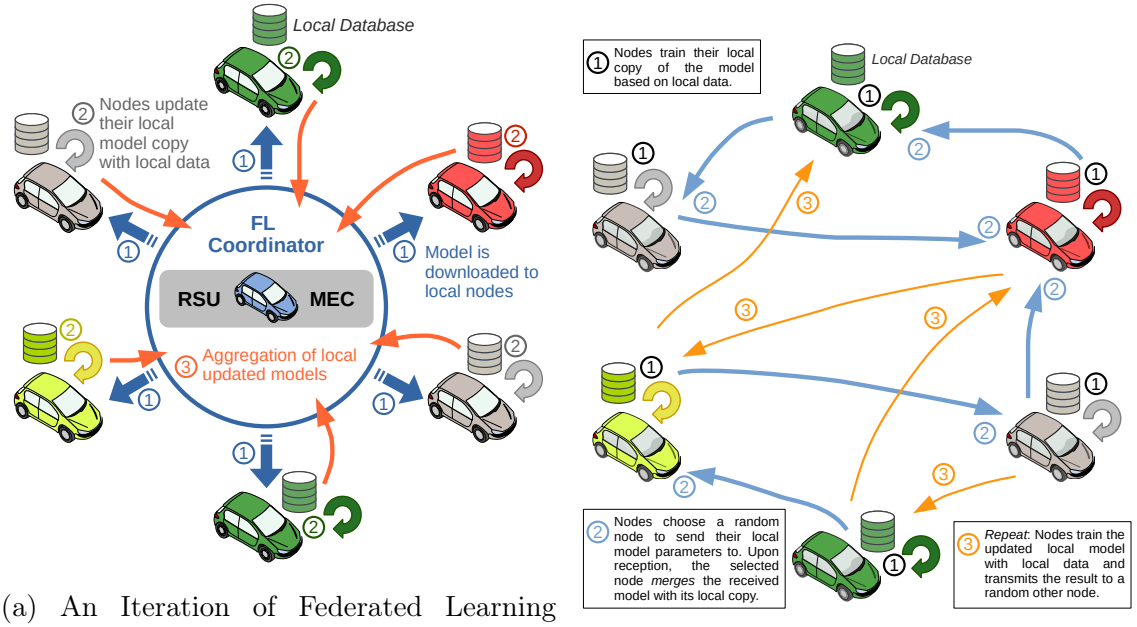
The **level 3** of the scale of knowledge distribution in vehicular networks, as defined in Figure 3.1, involves cooperative and distributed knowledge training. Namely, distributed knowledge training operations are emerging, and have been applied notably to vehicular networks. For example, in [65], vehicles use federated learning to cooperatively learn a model for electric vehicle range and energy demand prediction. Training a model in a distributed setting has several advantages, compared to gathering all training data in a centralized server for training, as described in [133]:

- When training a model in a distributed way, the training data can be kept on-board vehicles, thus avoiding legal privacy issues related with taking the data out of the vehicle to save it in a remote data center.
- Arguably, if the training data is large, keeping training data on board vehicles can reduce the bandwidth used in the network.

Algorithms for Distributed Knowledge Training In turn, several approaches have been developed to allow the distributed training of ML models. Distributed Machine Learning (DML) is an approach to train a ML model through multiple nodes which are distributed over a network [134]. It involves delegating the computation of a ML algorithm to remote nodes, using a single or distinct set of input data. To train ML models in a decentralized setting, nodes must cooperate and exchange training synchronization messages.

Federated Learning (FL) refers to the training of a ML model by multiple nodes owning a fragment of the training data [135]. Unlike DML, which splits data and model training as a means of increasing model training performance, FL is designed to train a model while keeping training data where it was originally obtained. It is designed to protect data privacy, as a model can be trained without taking the data out of each training node. The FL training procedure is decomposed into a sequence of training iterations. It involves two types of nodes, i.e., a centralized coordinator c, and a set of data-owning, potential training nodes S. The centralized coordinator c keeps a copy of the current state of the model being trained m_c. As illustrated by Figure 3.1a, each iteration involves the following operations:

1. A subset $S_{iter} \subset S$ of nodes is selected to perform local model training for this iteration.
2. The model m_c hosted by the coordinator, in its current state of training, is sent to each node in $s_i \in S_{iter}$, as shown in the step 1 of Figure 3.1a.
3. Each node $s_i \in S_{iter}$ locally performs a step of training on the received copy of m_c using the training data it owns, as illustrated by the step 2 of Figure 3.1a. The training is locally performed using traditional ML optimization techniques such as SGD, as described in Section 2.3.2.
4. Each node s_i sends back the parameters of the update of the m_c model, computed using local data, to the coordinator c. In turn, c updates the current



(a) An Iteration of Federated Learning Training (repeated from page 10)

(b) Principles of Gossip Learning

Figure 3.1: Centralized and Decentralized Forms of FL

value of m_c by aggregating the updates received from local nodes. This concludes the iteration of training, as illustrated by the step 3 of Figure 3.1a.

Gossip Learning (GL) is a fully decentralized alternative to FL, which eliminates the coordinator. Instead, the model is trained through direct V2V communications. As illustrated in Figure 3.1b, GL can be realized by local training nodes implementing the following two procedures:

1. *Local training and distribution:* Each local node s_i performs a local step of training of their own copy m_i of the model being trained, using local data. In turn, the updated parameters obtained for m_i are sent to a randomly selected remote node, which also participates in the GL training, as illustrated by the step 1 of Figure 3.1b.

2. *Local model merging:* When a local node s_j receives a model update from another node, it merges the received parameters with the parameters of its own copy of the model m_j, as illustrated by the step 2 of Figure 3.1b. In turn, it repeats a step of local training, whose result is sent to a random node.

In Multi-Agent Reinforcement Learning (MARL), distributed agents are cooperating to create knowledge as a policy describing how to interact with their environment in a way which maximizes a utility reward [136]. Other works have focused on the implementation of distributed versions of multiple variants of ML which were originally developed for centralized training [137, 138].

Knowledge Training Semantics The distribution of fully trained knowledge to vehicular nodes requires a semantic description of the input/output interface of the

knowledge, as well as the context in which it should be used. Yet, the definition of semantics for cooperative and distributed knowledge training is even more critical.

Like for the application of fully trained knowledge, in research papers which define distributed learning models, either (i) the considered topology is static, the training context of each node is known and constant, or (ii) the training context of each node is preset and known in advance.

Provided that no semantic is available to describe a distributed training process, a vehicle cannot understand:

- The interface of the model to train, i.e., the nature of the input data which is required to train the model, and the output content it should be associated to.
- The algorithm which is used for training, i.e., DML, FL, MARL, or another.
- The specific parameters of the training process, e.g., the definition of the layers of an ANN, the loss function to assess the efficiency of the model being trained, and other algorithm-specific parameters.

What is more, vehicular networks are highly decentralized networks in which the training environment of each vehicle changes dynamically, i.e., vehicles sense and discard new training data as they move. In these conditions, distributed knowledge training is highly unpractical as, in the vehicular medium, the training environment of each vehicle must potentially be reassessed at multiple occasions during the training. Namely, without exchanging semantic descriptions of the training context of models and the training environment of vehicles:

- How can a vehicle know which vehicles feature the right training context for participating in training, that is, the right types of input and output content, which potentially requires sensing in a highly specialized context?
- How can nodes of vehicular networks announce that a cooperative training process will be started, and describe the requirements for that process?

As such, a semantic description is critical both so that (i) nodes can communicate on their training contexts, and (ii) the required training context for a training process can be expressed, in order to be matched with the training context of nodes.

3.1.2 Existing Standards for Vehicular Knowledge Description

Standards have been defined and used for the semantic description of information in vehicular networks. As introduced in Section 1.1.2, ETSI-specified messages such as the CAM [5], DENM [6], or eventually CPM [7] provide a format to describe, respectively, vehicle kinematics, punctual road events such as accidents, or sensor-detected objects on the road.

What is more, specific ontologies have been defined for vehicular networks. For example, the Vehicle Signal Specification (VSS), as augmented in [86], defines a large set of vehicle-related information, such as the temperature of the engine of a vehicle. It features a data type, i.e., integer, a range of possible values, and a unit, i.e, -50 to 200 degrees Celsius, as well as a human readable description. Similar entries are provided for various elements of a vehicle.

As such, semantic descriptions for the mutually understandable and interoperable description of objects and quantities related to the vehicular environment are well advanced. In parallel, knowledge description standards exist as considered in Section 2.4, and can be used to implement some aspects of the semantic description required for knowledge.

As introduced in Section 2.4, the input and output interface of fully trained knowledge can be described using any of:

- Named information items from ontologies such as VSS.
- The OWL-S service description language [90].
- A WoT Thing description [91].

Yet, the description of the context to use or train a knowledge model, as well as the description of the objective of a model training is not directly documented in the aforementioned standards. As such, a mechanism of context semantic description must be defined for vehicular knowledge networking, based on the existing information and knowledge description standards.

In this section, we analyze the potential impact of knowledge description semantics for various levels of knowledge distribution in vehicular networks, as described in Figure 3.1. We find that semantic description is critical for all levels of knowledge distribution in vehicular networks, especially for cooperative knowledge training. Yet, providing knowledge description semantics is merely an initial step, and in turn must be integrated in knowledge dissemination and knowledge storage operations in vehicular networks, as we analyze in the following sections.

3.2 Analysis of Knowledge Dissemination

The semantic description of knowledge is critical to enable knowledge dissemination in vehicular networks, as described in Section 3.1. In turn, mechanisms must be developed which integrate semantics with a vehicular knowledge dissemination architecture. In this section, we analyze the potential of knowledge dissemination in vehicular networks and the open issues related to its implementation.

3.2.1 Dissemination of Fully Trained Knowledge

To illustrate the operations related to the dissemination of fully trained knowledge among vehicles of vehicular networks, which involves the *level 1 and 2* of the scale

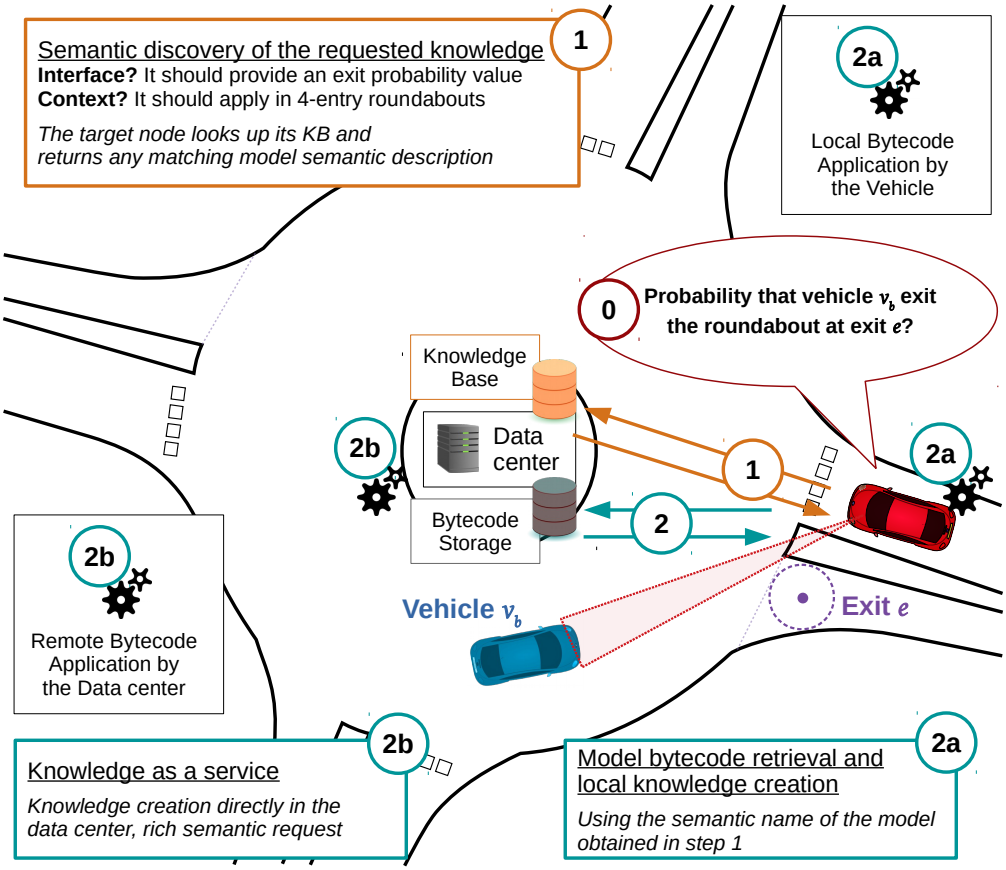


Figure 3.2: Overview of a Fully Trained Knowledge Dissemination Scenario

of knowledge distribution defined in Figure 3.1, we present an example scenario of the dissemination of a piece of knowledge.

As illustrated by Figure 3.2, we consider a model to assess the probability of a vehicle to exit a roundabout at the next available exit, based on its input kinematics data. Let us consider the following scenario: A highly automated vehicle v approaches a roundabout for the first time. It wishes to obtain knowledge about the probability of vehicles that are in the roundabout to exit, in order to safely decide whether to engage or not in the roundabout, as illustrated by the step 0 of Figure 3.2. Yet, v is not aware of a model to compute this probability, nor does it know if it exists. In turn, (i) v must disseminate a request to discover any existing knowledge which produces a roundabout exit probability as an output, as illustrated in the step 1 of Figure 3.2.

Let us consider the request has been routed to a node which knows the semantic description of a model, named m, which indeed produces roundabout exit probability as an output, (ii) the complete semantic description of the model m is returned to v, as illustrated in the step 1 of Figure 3.2. Through the semantic description, v is informed of the type of kinematics input it needs to provide to obtain the exit probability knowledge, as well as of a description of the contexts which are supported by the knowledge. For example, the model m must be applied on 4-entry roundabouts only.

Finally, if v is indeed crossing a 4-entry roundabout, i.e., in the right context to

use m, and it wishes to use this named knowledge, it has two options: (iii.a) request a remote node for a copy of the machine code of m, to directly use the knowledge when crossing the roundabout, as illustrated in the step 2a of Figure 3.2, or (iii.b) directly request a remote node the service to use m on behalf of v, provided input observations from v, as illustrated in the step 2b of Figure 3.2.

In turn, three aspects must be considered: knowledge discovery and knowledge use, either by requesting a copy of a model, or through *knowledge as a service*, where a remote node creates knowledge on behalf of another. We analyze the requirements for these aspects in further details in the following paragraphs.

Knowledge Discovery

Information is typically queried and referred to by its name in information-centric networking. The name of information itself can potentially be organized through specific naming schemes as described in Section 2.6.2. On the other hand, a knowledge model is identified and can be referred to through many aspects. It can be queried through:

- A name, e.g., chosen by the knowledge producer.
- All or a part of its input interface, e.g., One wants a model which takes a camera image as input.
- All of a part of its output interface, e.g., One wants a model which outputs roundabout exit probability knowledge.
- A filtering of its context of application, e.g., One wants all models which apply in 4-entry roundabouts.
- A combination of two or more of the previous aspects.

As such, knowledge discovery requests for vehicular networks may not only be based on a name, but also on rich contextual queries which allow vehicles to discover the available knowledge which closely matches their needs as well as their current driving context, i.e., the requested knowledge will be accurate in that context. In turn, if a matching knowledge is found, its full semantic description is returned to the query emitter.

As described in [139] which is a parallel work that surveyed the means of discovering services in IoT networks, knowledge queries could be routed in a decentralized approach. Alternatively, centralized entities gathering a list of the semantic descriptions of known knowledge models may be queried and reply with any match.

Knowledge Model Transmission

Once a vehicular node has issued a query for a knowledge model which matches its needs and driving context, a copy of the full semantic description of any matching

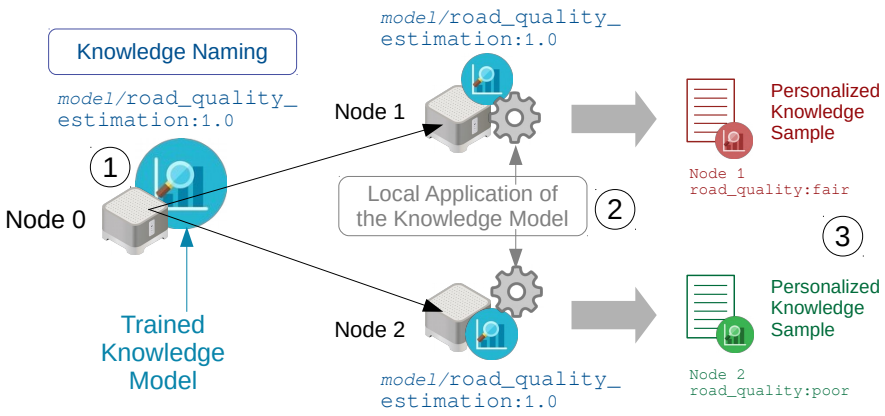


Figure 3.3: Knowledge Application By Static Model Distribution

model is returned, including a name m, input/output interface and the context for using the knowledge.

In turn, the vehicular node can use the full semantic description to request this time a copy of the machine code of the model which answers by the name m. In turn, the vehicular node can locally use the knowledge at any time, as it is in possession of a copy of both the semantic description of the knowledge, which describes how to use it, and its machine code, to actually execute the knowledge creation procedure.

Figure 3.3 illustrates the process of obtaining a copy of the machine code of a model whose name has been discovered through a query, for subsequent local use of the model.

Knowledge As a Service

Once a vehicular node has obtained the name m and the full semantic description of a model that it wishes to use, an alternative approach to requesting a copy of the machine code of the model is to request for *knowledge as a service*. We call *knowledge as a service* the application of a knowledge model by a remote node on behalf of another. Namely, instead of requesting for a copy of the machine code of the model m to apply it locally, a vehicular node can directly request a remote node to create the knowledge on behalf of it.

Namely, the requesting node gives the name m of the model to apply as a service as well as a set of well formed inputs. The request is forwarded to a node in possession of the machine code of m which creates the knowledge and returns the output to the request emitter.

Figure 3.4 illustrates the remote creation of knowledge using the *knowledge as a service* approach. The knowledge as a service approach is relevant notably in cases where the machine code of the model is very large, such as some deep learning models, given the limited storage space of vehicles. Similarly, models which are computationally intensive to use might be stored on data or edge servers with higher computational power than vehicles, and provide *knowledge as as service*.

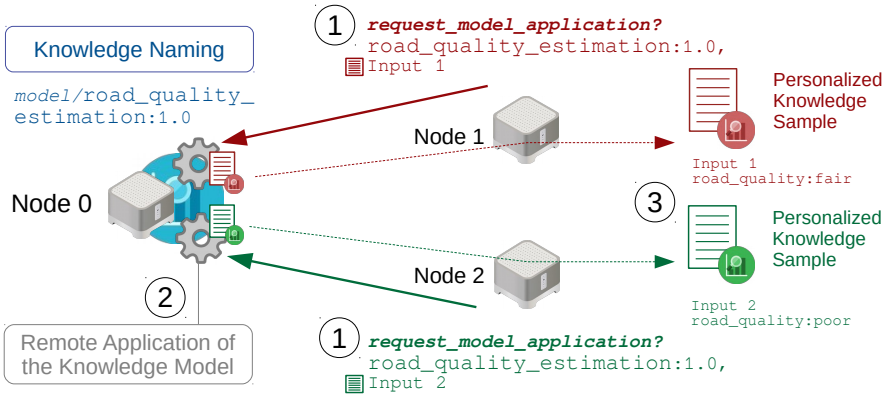


Figure 3.4: Illustration of Knowledge as a Service

Furthermore, specific knowledge dissemination mechanisms are required to support the orchestration of cooperative knowledge model training in vehicular networks, i.e., *level 4* of the scale of knowledge distribution, which we defined in Figure 3.1. In Section 3.2.2, we analyze the requirements and open issues for the orchestration of cooperative knowledge training in vehicular networks.

3.2.2 Orchestration of Cooperative Knowledge Training

Knowledge dissemination in the **level 4** of the scale of knowledge distribution defined in Figure 3.1 involves not only disseminating a fully trained model which was produced by a centralized entity. Rather, nodes have to be organized to cooperatively train a knowledge model by delegating training steps among them in a distributed manner. As such, the definition of model training semantics, as discussed in Section 3.1.1, is a key enabler to provide a foundation for vehicular cooperative knowledge training. Two aspects are critical for its architecture: (i) The initialization of a cooperative training process and, (ii) while a model is being trained cooperatively, the selection of vehicular nodes to contribute to the training at each training iteration.

On the one hand, the initialization of a training process may be performed by an arbitrary vehicular node, wishing to cooperatively train a new model. To initiate the training process, the node must provide a complete semantic description of the training task, including its interface, context of training, and training algorithm parameters. In turn, the training of new models has a potential to be agreed on and understood by nodes which have not been provided with a hard-coded knowledge of the parameters of the model to be trained. On the other hand, once the training task has been well defined through semantic description, the actual cooperative model training process must be orchestrated. Namely, the training is decomposed into iterations. At each iteration, a set of nodes must be selected to perform local steps of training, we call this process of selection of relevant training nodes the *orchestration* of the cooperative distributed model training.

The initiation and orchestration may be performed by a centralized controller, as in FL, which directly selects a set of training nodes to produce and return model

Table 3.1: Existing Training Node Selection Mechanisms in FL Training

Category		Approach	Reference
Resource Constraints	Network Channel Quality	Link quality-aware training node selection	[140]
		Irrelevant updates discarding mechanism	[141]
		Compression mechanisms	[142, 143]
	Computational Power	Computation-aware training node selection	[140]
	Energy Consumption	Energy-aware training node selection	[144, 145]
Training Data Distribution		Heterogeneous data distribution learning	[146, 147]
		Data distribution-based clustering of training nodes	[148]
		Global and local model feature fusion	[149]
		Partial data sharing or augmentation [150]	[151, 152]
Security Concerns		Protection against forged model updates	[153]
		Privacy of training input	[154]
		Reputation-based training node selection	[155]
Incentive Mechanisms		Fair distribution of monetary costs and rewards for training nodes	[156, 157]
Optimal convergence of the distributed gradient descent algorithm			[158, 159]

updates at each iteration. Alternatively, it may be fully decentralized, as in GL, with vehicles directly exchanging their model updates in a V2V approach. In this section, we analyze the key needs and existing issues for the orchestration of **level 4** cooperative knowledge training processes, given the specificity of the vehicular environment, while remaining agnostic to the actual architecture of the training, i.e., using a controller or fully decentralized.

Existing Orchestration Approaches

Table 3.1 summarizes the existing approaches to orchestrate the selection of training nodes at each iteration of a FL cooperative training process. Namely, mechanisms have been defined which belong to the following broad categories:

- In *resources constraints-based* orchestration, nodes are selected at each training iteration based on their current resources availability. Typically, approaches have been considered in mobile environments to select nodes for training based on their channel quality, available computational power, or energy level.
- In *training data distribution-based* orchestration, the distribution of training data is taken into account when selecting training nodes. Typically, in cases where the data is non independent and identically distributed among nodes, mechanisms of clustering of training nodes or partial data sharing can be implemented.
- In *security-based* orchestration, the training nodes which are the most trusted and the least likely to pose a security threat are selected for each training iteration. One aspect is the detection of forged model updates. A potential solution to implement security-based orchestration is to define a level of reputation for each node.
- Finally, miscellaneous types of orchestration have been studied, such as the definition of training incentives to motivate nodes to participate in a training process. In turn, the nodes which have agreed to join the cooperative model

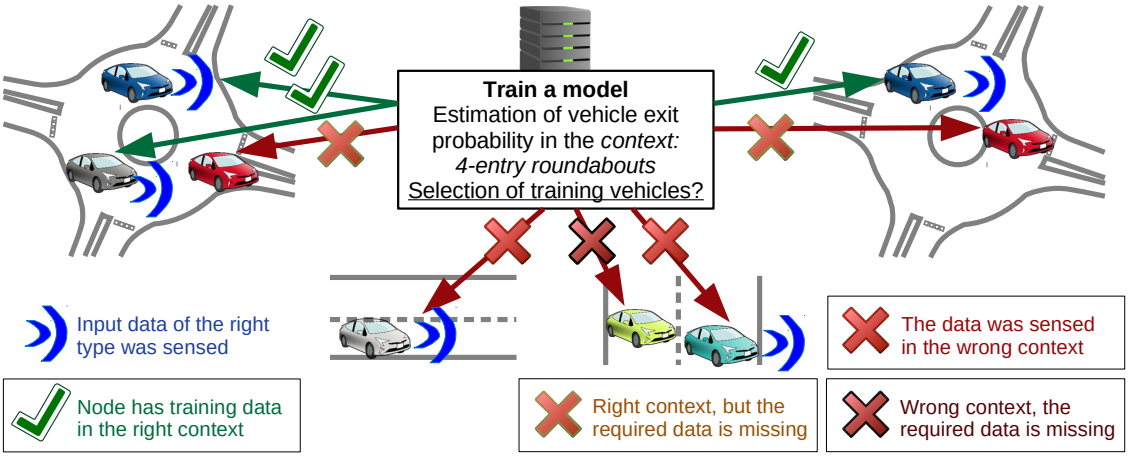


Figure 3.5: Illustration of Context-Aware Cooperative Training Orchestration

training can be selected. What is more, some approaches have been defined to select nodes in order to maximize the performance and the convergence of the underlying learning algorithms such as distributed gradient descent, as in [158] or [159].

The Critical Aspect of Context-Based Orchestration

Yet, in the existing FL cooperative model training approaches, all training nodes are assumed to feature the right training environment for the considered model. For example, let us consider the cooperative training of a model to estimate the probability of a vehicle to exit a roundabout at the next exit, based on its position and kinematics in the roundabout. As illustrated by Figure 3.5, in order to train a model cooperatively, the training task must first be initialized and accompanied with well-defined training semantics, e.g., we want to train a model to produce the output 'roundabout exit probability' from the inputs 'heading', 'lateral position', and 'distance to the next exit' in the roundabout. What is more, the training data should be collected in the following context: 4-entry roundabouts with a radius of more than 20m. Then, nodes are able to match their current training environment with the required context, i.e., whether they are currently sensing data from vehicles in a 4-entry roundabout with a radius greater than 20m. In turn, they can or cannot take part in the training of the given model.

Yet, to the best of our knowledge, in the existing approaches of FL model training orchestration, the training environment of each vehicle is not considered. Namely, either:

- The study is run on a single machine as a proof of concept for the model.
- The study considers static nodes whose training environment is known in advance as they do not move.
- A set of training nodes with the right context is known in advance.

- In real deployment to mobile networks, proprietary mechanisms are enforced to select training nodes, which *de facto* use data which were collected in the intended context.

Nonetheless, we envision future applications of cooperative knowledge training, where nodes can create and discover new training tasks on the fly, which were not manually implemented into the computing units of selected vehicles belonging to the same fleet of vehicles. In that case, the semantic description of the training task, to describe the objective of the training, as well as the training environment which should be used, is critical to make sure that a node is the right node for training a model.

A key question for a training node is: Am I the right node to train this specific model? Regardless of the optimizations for the selection of training nodes described in Table 3.1, if a node does not possess the right type of data for training, or if it does not feature the right training environment, i.e., its data was collected in the wrong context, then it cannot participate in the training task.

In parallel, vehicular networks are highly complex architectures in which potential training nodes move all the time, as such, their training environment is highly volatile, and they may join or leave the training process at any time. In turn, the current algorithms of training orchestration, which assume a valid training context, cannot function in these conditions.

As such, it is critical to define a framework to orchestrate the training node selection for cooperative knowledge training in the complex vehicular environment where the training context of vehicles is highly dynamic. Prior to resources or data distribution-based optimizations of training node selection as described in Table 3.1, semantic-based mechanisms must be defined for nodes to exchange about their training environment, in order to ensure that the selected nodes are in the right context to train the right model.

3.2.3 Suitability of the Existing Networking Approaches

In this section, we investigate the suitability of existing networking paradigms to support the semantic-based knowledge dissemination approaches, which we introduced in Sections 3.1 and 3.2.

IoT Approaches

As introduced in Section 1.1.2, IoT architectures involve an IoT platform which receives information from vehicular nodes. In turn, the information can be exposed to nodes through smart applications which are using the input data from the vehicular nodes. The communication between the IoT platform and the vehicular nodes is supported by messaging protocols such as MQTT [9], CoAP [10], or AMQP [11].

In turn, as illustrated in Figure 3.6, the implementation of the distribution of knowledge requires:

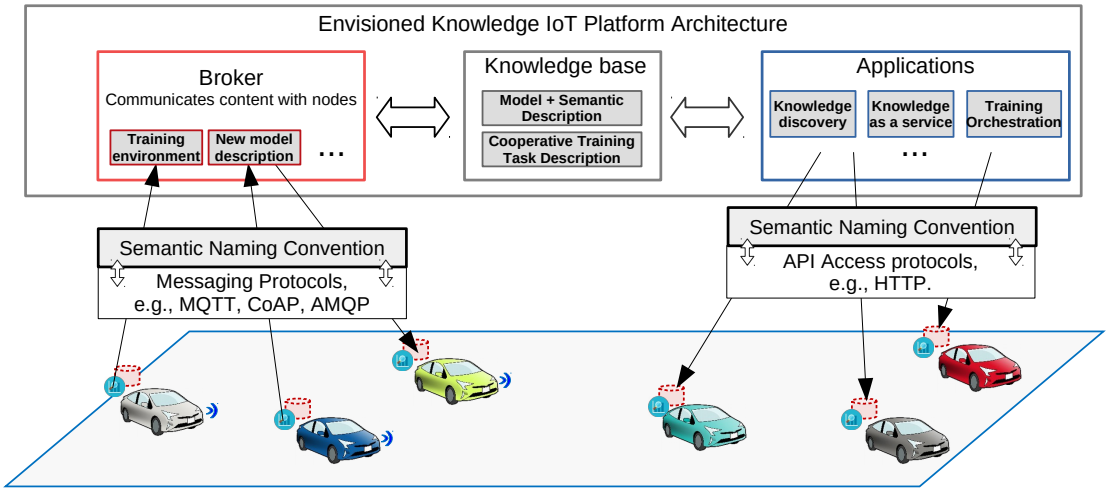


Figure 3.6: Analysis of the Requirements for IoT Vehicular Knowledge Networking

- The definition of common semantic rules to describe models and model training tasks, to be used by both the IoT platform and the vehicular nodes.

- The integration of a knowledge base in the IoT platform containing the semantic description and machine code of (i) known fully trained models, and (ii) models being trained cooperatively by nodes under the supervision of the IoT platform.

- The adaptation of a naming convention for messages exchanged through messaging protocols, e.g., MQTT, and application APIs, to integrate knowledge semantics.

- The development of specialized applications for specific tasks, such as the coordination of knowledge training, the discovery of knowledge in the knowledge base of the IoT platform, or *knowledge as a service* applications.

Information-Centric Approaches

As described in Section 2.6.2, ICN uncouples content from its location and lets nodes directly issue requests for named information. NDN is a popular implementation of ICN, which identifies information through a URI-like hierarchical name. In a NDN architecture, nodes can be producers and/or consumers of a named content.

In turn, as illustrated by Figure 3.7, the implementation of knowledge dissemination operations using ICN requires the following points:

- To begin with, a naming convention must be designed to convey the semantic annotations of knowledge in ICN packet names. By encapsulating the semantic annotations of knowledge in the URI-like hierarchical naming scheme of NDN, (i) knowledge discovery requests, (ii) full model retrieval, or (iii) knowledge as a service can be conveyed over NDN.

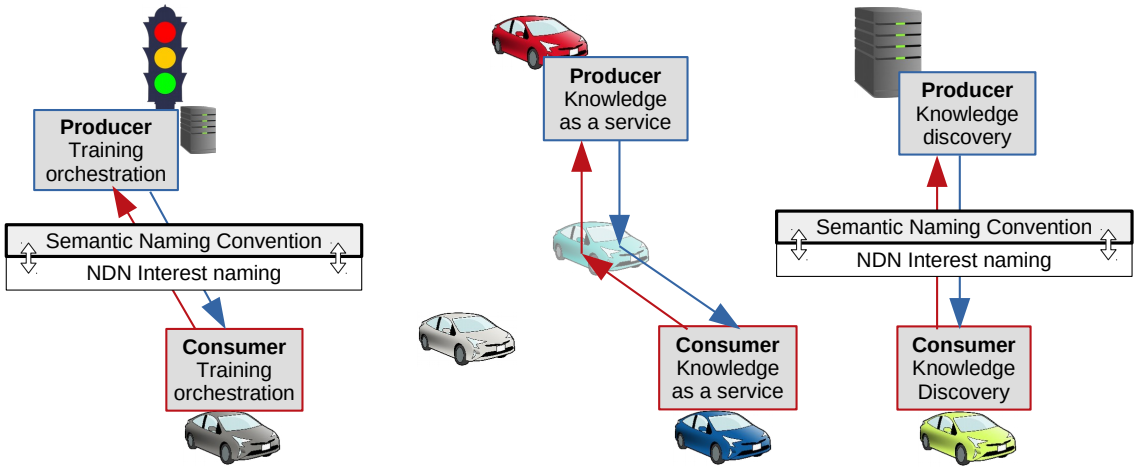


Figure 3.7: Analysis of the Requirements for NDN Vehicular Knowledge Networking

- In turn, the implementation of applications from producer nodes to implement the (i) knowledge discovery requests, (ii) full model retrieval, or (iii) *knowledge as a service* tasks must be realized. For example, [160] investigated on the relevance of NDN as-is to support the networking of knowledge as deep learning models. NDN can be used to some extent to support remote knowledge creation with local input using the hierarchical name structure. For example, given a knowledge model of name `/road/traffic` able to predict the traffic density from a location and time, a remote application request could be expressed as: `/road/traffic/road-`r`/tomorrow-8am`.

Yet, it is not practical to encapsulate complex content in a NDN packet name, such as a camera image bitmap input data for knowledge as a service. What is more, the definition of a format to convey complex knowledge semantics, e.g., knowledge training semantics, within ICN/NDN content names has not been defined, to the best of our knowledge.

Function-Centric Approaches

FCN integrates the execution of named functions on named information items, as described in Section 2.6.3. As such, FCN approaches provide a native support of *knowledge as a service* operations, provided that the nodes which request the application of the function know its name beforehand. However, function networking is a weaker form of knowledge networking, as it does not integrate context annotations, and there is no means of requesting for the application of a function in a given context. What is more, the orchestration of cooperative training tasks is beyond the feature set of NFN.

Regardless of the architecture which is chosen to implement vehicular knowledge networking and dissemination, three key elements are required, as analyzed in Sections 3.1 and 3.2: (i) The definition of knowledge description semantics, (ii) their integration in existing messaging and communication protocols, to convey semantically-rich and contextual requests of knowledge discovery, application or training environment discovery, and (iii) based on the integration of semantics in

messaging protocols, the definition of applications to implement knowledge discovery, knowledge as a service, or knowledge training orchestration.

3.3 Analysis of Knowledge Storage

The storage of knowledge is a complimentary aspect to the dissemination of knowledge. In this section, we analyze the needs and the open issues for the implementation of knowledge storage in vehicular networks, which we put in relation to our knowledge description and knowledge dissemination analyses in Sections 3.1 and 3.2. Knowledge storage aspects are considered at two levels, namely, the choice of database architectures and the choice of storage locations.

3.3.1 Database-level Requirements

As introduced in Section 2.7, a key difference between information and knowledge storage lies in the scope and lifetime of each content. In many safety applications, the information is short-lived and must be delivered with very low delay. This is, for example, the case of kinematics data exchange through ETSI CAM [5] messages. In turn, the ETSI LDM [96] database is optimized for fast insertion and extraction of information.

On the other hand, knowledge is trained to produce abstract outputs in a specific context. As such, knowledge is valid as long as it is applied in the context it was built for. Rather than optimizing for very-low-delay insertion and extraction in a database, the storage of knowledge should be efficient in storing and interpreting the semantic contextual annotations which define it, as described in Section 3.1. Namely, a key asset of a knowledge base is to be efficient at answering rich semantic queries to discover or retrieve knowledge based on a partial semantic description, e.g., using the SPARQL [88] query language. As such, relational-type databases could be more adapted for knowledge storage than information-optimized databases such as the ETSI LDM.

3.3.2 Storage Location and Caching Requirements

Mechanisms have been defined to cache information where it is likely to be requested by vehicles. In turn, it can be retrieved by vehicles through direct V2V or V2I communications, which is typically faster than sending a request to a remote data center. For example, [49] described a scheme to cache content in road side units close to vehicles, based on vehicular mobility, such that it can be retrieved efficiently by vehicles.

Similarly, knowledge can be stored and cached close to the vehicles in key locations where it is likely to be needed. In turn, the retrieval of full copies of knowledge models or their use through remote *knowledge as a service* can be performed faster. As such, a key challenge of knowledge storage is the choice of where copies of a spe-

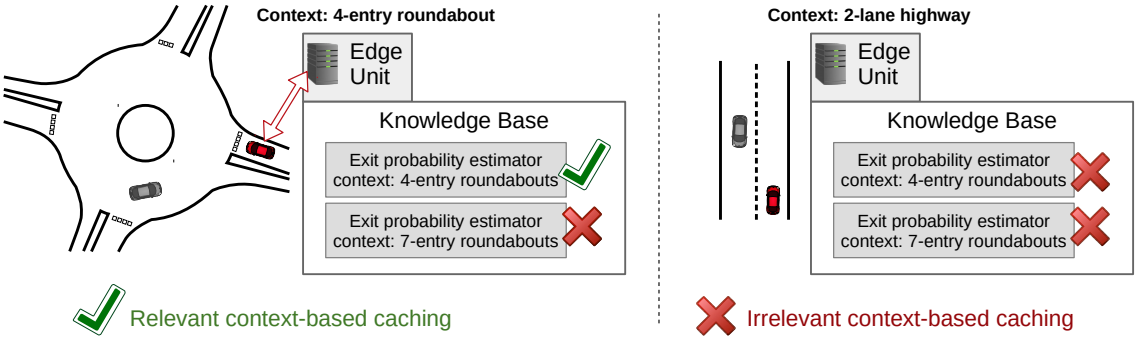


Figure 3.8: Illustration of Context-Aware Knowledge Caching Mechanisms

cific knowledge model should be stored and cached. For example, let us consider a model which can estimate the probability of vehicles in a roundabout to exit at their next available exit in a context of 4-entry roundabouts. As illustrated in Figure 3.8, it makes sense to cache a copy of this knowledge model in road side units and edge servers next to such 4-entry roundabouts. In turn, the vehicles which drive by can request a copy of the model, or a service of knowledge creation to the edge unit, and obtain a fast direct response. On the contrary, it does not make sense to store the knowledge close to a 5-entry roundabout, and even less in a road side unit on a highway.

As such, as with knowledge dissemination, the integration of a semantic description of knowledge models into storage operations is critical. Namely, the description of the context of application must be understood by a caching agent, to detect the locations which match that context on the map, and cache the knowledge. Similarly, the orchestration of cooperative model training, at least in the case where a centralized controller orchestrates the training as in FL, may be hosted in nodes which are close to the areas which feature the right training environment.

Table 2.2 described the existing approaches to cache content in vehicular networks, based on (i) vehicular mobility, (ii) content popularity, (iii) interest and social groups of vehicles, (iv) channel quality, or (v) predictions of the needs of vehicles. These approaches are pertinent for information and can be adapted for knowledge. Yet, to the best of our knowledge, no caching mechanisms have been defined which take a generic semantic description of knowledge into account, to cache knowledge close to its meant context of application or training. As such, like for knowledge dissemination, a framework is required to integrate and organize knowledge storage based on description semantics.

3.4 Problem Formulation

In current vehicular works, the distribution of knowledge is performed by assuming that nodes are *a priori* aware of the nature of the knowledge to be used, trained, or shared. In turn, this raises scalability and interoperability issues, as (i) in realistic environments, on-board units of vehicles must be hard-coded to adapt to each knowledge model they require to use, (ii) knowledge cannot be shared with nodes which have not received the hard-coded description of its interface and context of

application. As a consequence, knowledge is segmented by training entities and automakers. While in some cases it is an intentional choice made by automakers to keep a model proprietary, in other cases it limits the reach of existing knowledge, and requires independent training of similar models, which represents a significant excess cost compared to using and contributing to existing knowledge.

As such, associating knowledge models with semantics to describe their interface and context of training and application is a necessary condition to extend the reach of knowledge networking in vehicular networks, and power cooperative intelligent applications relying on knowledge dissemination and storage. Current architectures which were optimized for information distribution must be adapted and extended to support the distribution of not only trained models but also semantic descriptions of the models, allowing new vehicles to discover and take part in *knowledge as a service* applications, or the cooperative training of a new model.

In this doctoral work, we investigate the impact of integrating a semantic description of knowledge to knowledge distribution in vehicular networks:

- Knowledge description semantics should be defined and tightly-coupled with knowledge building algorithms and executable code, as a prerequisite to allowing knowledge networking in vehicular networks.
- In turn, we define knowledge dissemination mechanisms, which take the intended context of use and context of training of a model into account, and investigate whether they improve the efficiency of knowledge distribution aspects, as well as the accuracy of the produced knowledge.

In Chapter 4, we define and theoretically evaluate a Vehicular Knowledge Networking framework, which integrates knowledge models and knowledge description. In turn, we provide an overview of the performance of the framework through two complementary applications. On the one hand, we investigate the impact of the knowledge distribution framework on network efficiency metrics. On the other hand, we investigate the impact of knowledge networking on the accuracy of the exchanged knowledge. These two preliminary applications illustrate the potential benefits of VKN. In turn, in Chapters 5 and 6, we define a realistic knowledge model and evaluate the performance of its context-aware distribution in a setting of realistic vehicular mobility.

Chapter 4

Definition of a Vehicular Knowledge Networking Framework

In this chapter, we define a conceptual framework to integrate knowledge description semantics to knowledge creation, storage, and distribution in vehicular networks, which we call the Vehicular Knowledge Networking (VKN) framework. After introducing the overall architecture of the framework, we analyze the potential gains of performance associated with the use of VKN for use cases of knowledge networking, namely (i) knowledge as a service, i.e., remote knowledge application and (ii) cooperative knowledge training through federated learning. This analysis is a prerequisite to demonstrate initial and preliminary performance gains, before implementing knowledge networking on a realistic vehicular application in the next chapters.

4.1 Architecture of the Framework

4.1.1 Motivation

As analyzed in Chapter 3, it is highly challenging to perform operations of knowledge networking in the vehicular environment. As the topology is dynamically changing due to the mobility of vehicles, the driving context of vehicles is constantly evolving. This impacts the ability of vehicles (i) use existing models or (ii) take part in the cooperative training of a model. In turn, there is a need for coherence in the distribution of knowledge, based on the fast-evolving driving context of each vehicle. For example, a knowledge model which helps crossing a roundabout, trained for 4-entry roundabouts should not be used by a vehicle crossing a 7-entry roundabout, and should not be stored in irrelevant locations, e.g., in a edge server close to a signalized intersection.

Moreover, we defined a scale for the distribution of knowledge in vehicular networks in Figure 3.1. Namely, the operations of knowledge distribution are varied and may require different levels of cooperation, from no cooperation when a model is designed for the exclusive use of a node, to cooperative distribution of fully trained

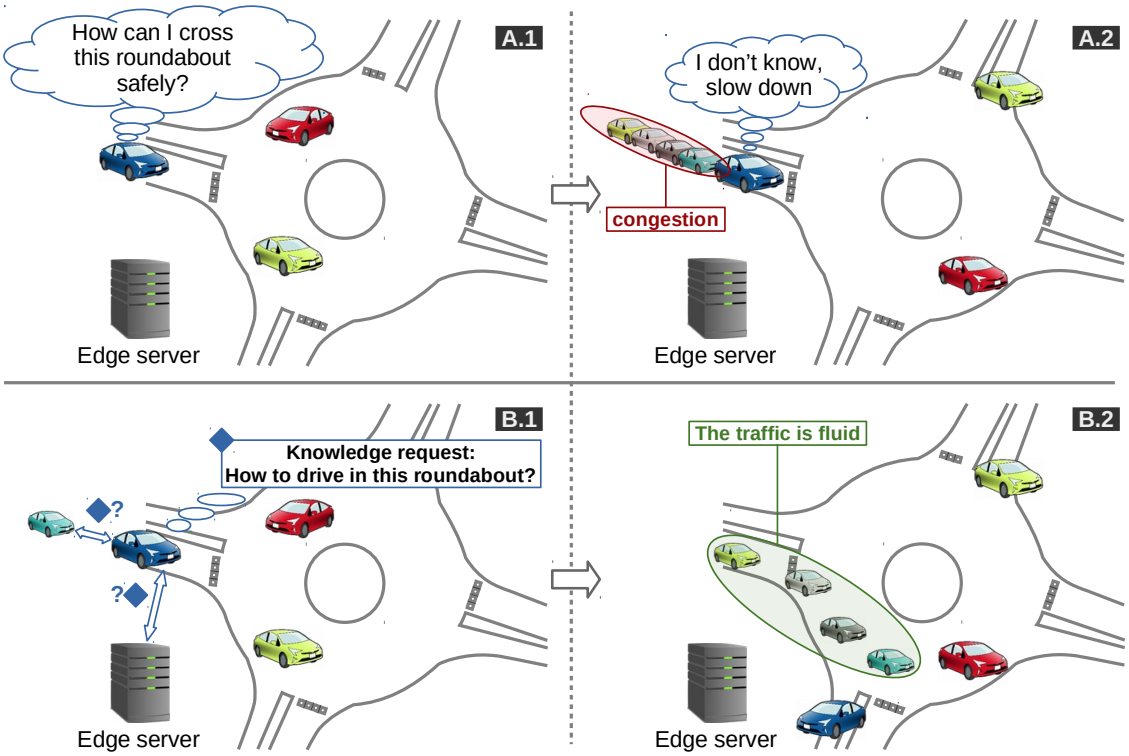


Figure 4.1: Motivation for the Concept of a Vehicular Networking Framework

knowledge, to the cooperative training of knowledge.

In turn, it is critical to define (i) knowledge description semantics which describe both the input/output interface of a model and its necessary context of use or training, integrated with a framework to organize a coherent (ii) storage and (iii) dissemination of knowledge for each level of knowledge distribution, as defined in Figure 3.1.

Figure 4.1 illustrates the motivation for the definition of a framework to integrate knowledge distribution in vehicular networks, in the case of the distribution of fully trained models. As shown in the point `A.1` of the figure, a dark blue-colored highly automated connected vehicle reaches a roundabout in a context which is not covered by its self-driving algorithms, e.g., the vehicle has knowledge to navigate in a USA-based roundabout, but is crossing a European roundabout for the first time. In turn, it does not know how to cross the roundabout both efficiently and safely. Without a knowledge distribution framework which lets the vehicle request for specific knowledge on how to cross this type of European roundabout, the vehicle slows down and produces a traffic congestion, as illustrated by point `A.2`.

On the contrary, provided with a framework to express its need for knowledge using rich describing semantics, and forward it to other nodes such as another vehicle or an edge server, the vehicle can obtain knowledge on how to cross this type of roundabout safely, as shown in point `B.1`. Finally, in point `B.2`, the vehicle uses the received knowledge to cross the roundabout without slowing down unnecessarily, and the traffic congestion is avoided.

As described in Section 2.4, vehicular clouds or edge computing are support-

ing technologies on top of which knowledge storage and dissemination could be implemented [102]. Nevertheless, a comprehensive framework, that can be implemented on top of existing architectures, is required to define knowledge description semantics as well as applications which use the semantic for knowledge storage and dissemination.

In this section, we define and describe the architecture of a framework for knowledge distribution in vehicular networks, which we name *Vehicular Knowledge Networking* (VKN). As part of VKN, we use and define means of semantic knowledge description, which then allow the implementation of knowledge storage and dissemination operations. In turn, it can be implemented on top of existing content networking architectures to enable efficient knowledge networking in vehicular networks.

We contribute the concept of a knowledge networking framework architecture for vehicular networks, which further details the knowledge description, storage, and dissemination architecture. While the generic concept of a knowledge-centric framework has been defined in [125], the definition of (i) the VKN framework and of (ii) the concept of KCN in [125], have been performed in parallel and not in sequence, despite the fact that the KCN concept [125] has been published earlier.

What is more, as part of VKN, a key contribution is the physical separation of the semantic description of knowledge models from their bytecode, i.e., the machine code which actually produces output from well-formed input.

4.1.2 Knowledge Description

As a main contribution, we define a VKN framework which is divided into three main parts, namely, (i) knowledge semantic description, (ii) knowledge storage, and (iii) knowledge dissemination. In this section, we describe the knowledge semantic description aspect of the framework.

Classification of Knowledge

To begin with, we contribute a taxonomy of knowledge in vehicular networks, and the general forms it can take. As illustrated by Figure 4.2, we separate knowledge in vehicular networks into three aspects, which we name (i) knowledge models description, (ii) knowledge models bytecode, and (iii) knowledge samples:

- Knowledge model descriptions are a semantic description of the input/output interface as well as the context of application or training of a knowledge model. It is a machine-understandable description of the metadata of the considered knowledge model. It includes a name and version code for the model, the list of input and output items, and their range of possible values, as well as preconditions for the application of the knowledge model to produce new output, i.e., the required context of use of the model.
- Knowledge model bytecodes refer to the actual machine code associated with a

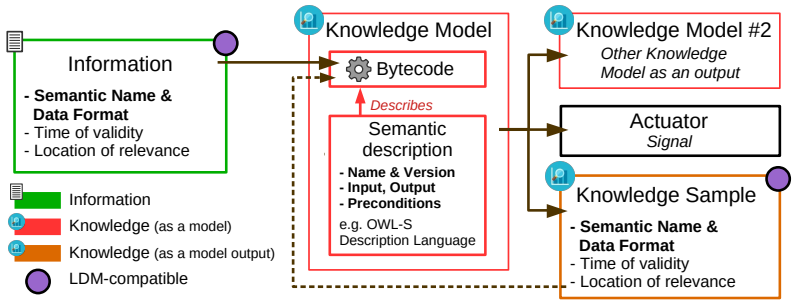


Figure 4.2: The Vehicular Knowledge Ecosystem

knowledge model description, which produces an output based on well-formed input. Knowledge models might be implemented using various algorithms, programming languages, and execution environments. As such, two models with the same set of input and output parameters may feature different bytecodes.

- Knowledge samples refer to the structured output produced by a knowledge model. It is structured like information, but called knowledge as it is an abstract content produced by a model.

While the bytecode of a model produces the actual output values from well-formed input, the semantic description of a model is critical to ensure that the model can be used with the right interface and in the right context by a node which was not provided with a hard-coded knowledge of the interface of the model.

We separate knowledge models and knowledge samples in order to match the different understanding of knowledge, as introduced in Section 2.3.1. In a strict definition, knowledge only refers to knowledge models. In more liberal definitions, the abstract content which is produced by a knowledge model can also be called knowledge, even though it is structured like information and can be exchanged as such.

Figure 4.2 is generic and does not imply the use of any specific algorithm or technology to implement knowledge models. It illustrates the general relationship between (i) information, (ii) knowledge model descriptions, (iii) knowledge model bytecodes, and (iv) knowledge samples.

- On the left side of the figure, in a dark green-colored box, information is stored on-board connected vehicles in the LDM database, as introduced in Section 2.5. As such, it is associated with a semantic time and area of validity, as well as a semantic data format, described as part of existing standards such as CAM [5] or DENM [6].

- On the center of the figure, the separation between the semantic description of a knowledge model and its bytecode is illustrated. The semantic description of the model includes a name and version code, as well as a description of the input and output interface of the model, and its context of use, i.e., *preconditions*.

- On the right side of the figure, the relationship between 'knowledge models' and 'knowledge samples' is illustrated. A knowledge model may output (i) other knowledge models, (ii) actuating signals to react to a given input with an action, and (iii) *knowledge samples*, which feature a time and area of validity. Namely, knowledge samples are structured similarly to information, despite being a piece of abstract content produced by the analysis of information. As a consequence, they can be used as input to another knowledge model and generate further composed knowledge.

Implementation of Knowledge Model Semantic Descriptions

To illustrate the concepts of *knowledge model*, *knowledge sample*, *semantic description* and *bytecode* introduced in Figure 4.2, we provide an example applied to a hypothetical model which predicts passenger comfort on-board a connected vehicle based on environmental factors.

As such, we define a semantic description of two hypothetical models which we name `model.env_comfort` and `model.fetch_driving_behavior`:

- A first model is named `model.env_comfort`. As a general description, it aims at predicting a discrete level of passenger comfort in a target vehicle, from the following named input items types:
  - `Road.Traffic`: Defined as a discrete indicator of the level of traffic congestion surrounding the target vehicle ∈ `[FLUID, CONGESTED]`.
  - `Road.Visibility`: Defined as a discrete indicator of the level of visibility around the target vehicle ∈ `[CLEAR,OBSTRUCTED]`.
  - `TwoWheelers.Concentration`: Defined as a discrete indicator of the concentration of two-wheelers around the target vehicle ∈ `[HIGH,MEDIUM,LOW]`.

  In turn, it produces the following output content type:

  `Road.ComfortLevel`: Defined as a discrete indicator of the level of passenger comfort on-board the passenger vehicle ∈ `[GOOD,FAIR,POOR]`.

  The `model.env_comfort` has been designed to be used only in the context of driving in a specific `State`, which is `State = California`.

- A second model is named `model.feth_driving_behavior`. Its aims is to look up a centralized database of models to return a semantic description of a model which lets vehicle adapt their driving, given an input `Road.ComfortLevel` ∈ `[GOOD,FAIR,POOR]` and `Location.Town` which is a unique name for a town.

Figure 4.3 illustrates the relationship between the two aforementioned models, considering the classification of knowledge in vehicular networks which we defined in Figure 4.2. On the top of the figure, the example knowledge model which we named `model.env_comfort` is introduced. It produces a `Road.ComfortLevel` output knowledge sample, which is in turn given as input to the `model.fetch_drivin g_behavior` model, which outputs a description of a model to assist self-driving.

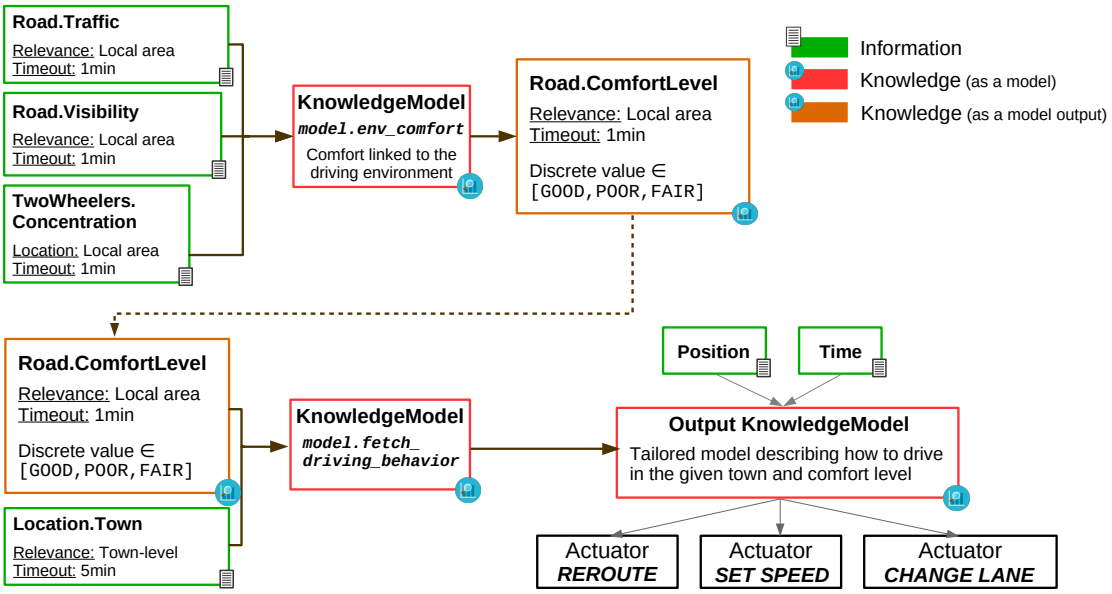


Figure 4.3: Knowledge Models for Passenger Comfort Handling

To implement the semantic description of knowledge models, a specific language is required. Based on the analysis in Section 3.1, we define the following description items as part of VKN:

- Model metadata, i.e., a name and version code for a model.
- A list of input and output named items to describe the interface of the model.
- The description of a context in which the model can be used.

As such, we select the OWL-S [90] service description language to implement the semantic description of models as part of the VKN framework which we contribute. As introduced in Section 2.4. The *process model* standard of OWL-S can be used to describe the set of inputs, outputs, and preconditions of a model. Figure 4.4 provides an example OWL-S description of the `model.env_comfort` model which was introduced in Figure 4.3. The context of use of the model is described using the Knowledge Interchange Format (KIF) first order logic [161], which in this case lets us define the condition that the `State` of use of the model should be *California*.

To illustrate the concept, Algorithm 1 provides an example simplified implementation of the bytecode of `model.env_comfort` written in pseudocode. In realistic applications, it could be substituted with a more complex AI-based model, as VKN is a general framework which supports any AI-based knowledge with an input and output interface.

Namely, we stay generic in the definition of the VKN framework and do not make the explicit choice of a technology to implement bytecodes. Yet, to provide a common as well as secure environment and format for the execution of bytecode received from remote nodes, we suggest the use of virtualization containers, as they provide both flexibility of implementation, portability, and isolation which improves security compared to running the code directly on the on-board unit system of the vehicle. An overview of virtualization containers is described in [162].

```
<AtomicProcess ID="model.env_comfort:1.1">
  <hasInput resource="#traffic" />
  <hasInput resource="#visibility" />
  <hasInput resource="#twoWheelers" />
  <hasOutput resource="#comfort">

  <!-- As part of OWL-S, we indicate the items which are required
   to describe the context of use of the model as an input -->
  <hasInput resource="#state" />

  <!-- Use a KIF first order logic expression to describe the
   required context of application 'state == California' -->
  <hasPrecondition>
    <KIF-Expression>
      <expressionBody>
        (= ?state "California")
      </expressionBody>
    </KIF-Expression>
  </hasPrecondition>

</AtomicProcess>

<Input ID="traffic">
  <parameterType resource="#Road.Traffic" />
</Input>
<Input ID="visibility">
  <parameterType resource="#Road.Visibility" />
</Input>
<Input ID="twoWheelers">
  <parameterType resource="#TwoWheelers.Concentration" />
</Input>
<Output ID="comfort">
  <parameterType resource="#Road.ComfortLevel" />
</Output>
<Input ID="state">
  <parameterType resource="#State" />
</Input>
```

Figure 4.4: Comfort Model Description Structure in OWL-S

Algorithm 1 Simplified Algorithm to Compute Comfort from Environmental Parameters

Input
$c_{tw} \in$ [HIGH, MEDIUM, LOW]
$v \in$ [CLEAR, OBSTRUCTED]
$tr \in$ [FLUID, CONGESTED]
Output
$cft \in$ [GOOD, FAIR, POOR]

```
if c_tw = LOW v = CLEAR tr = FLUID then
    cft ← GOOD
else if c_tw = HIGH then
    cft ← POOR
else
    cft ← FAIR
end if
```

4.1.3 Knowledge Storage

In the second part of VKN, namely, knowledge storage, we define an architecture for knowledge storage using knowledge bases in vehicular nodes.

Knowledge Base Integration

As analyzed in Section 3.3, the existing LDM database designed for information storage on-board vehicles is optimized for fast insertion and extraction time of information. This can be explained by the very low latency which is required for information access in some safety applications, such as the exchange of kinematics information through ETSI CAM [5].

On the other hand, the relevance of knowledge models depends on the context in which they are applied, rather than their freshness. In turn, knowledge bases should be optimized to answer rich queries to discover knowledge based on parts of its semantic description, e.g., using the SPARQL querying language, rather than to provide the latest inserted knowledge with very low delay. As such, there is a need to define a knowledge base on-board vehicles which is designed to answer rich context-based queries to organize the storage and retrieval of knowledge based on its semantic description characteristics.

Then, as a contribution and as part of the knowledge storage aspect of VKN, we define a knowledge base layer between the ETSI facilities layer, containing the LDM, and the applications on-board a connected vehicle, as illustrated by Figure 4.5. The facilities layer contains the LDM which can store information items with an area and time of validity. As such, it can be extended to store *knowledge samples*, which are structured like information as described in Section 4.1.2. The knowledge layer is placed between the facilities and the application layer of the on-board unit of the connected vehicle. It involves two independent storage units: (i) a knowledge model bytecode storage and (ii) a knowledge model semantic description storage.

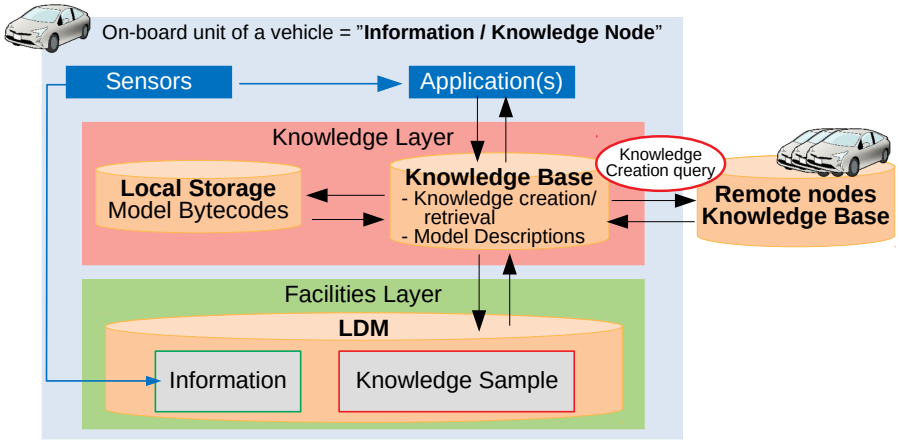


Figure 4.5: On-Board Storage of Knowledge and Information

Separation of Knowledge Model Description and Bytecode

As a strong choice and contribution as part of the VKN storage architecture, we physically separate the storage of knowledge model bytecodes, and their associated semantic descriptions. In turn, a knowledge model description may be stored in a node without the corresponding bytecode. The semantic description is a lightweight content which can be shared with multiple nodes, whereas the bytecode is typically a larger file.

In Figure 4.5, the knowledge model bytecode storage is shown in the 'Local Storage' database. It is interconnected with the knowledge model semantic description storage in the 'Knowledge Base' database. The bytecodes stored in the local storage are uncoupled from the model descriptions in the KB, i.e., some model descriptions might be stored locally without a corresponding bytecode.

In turn, knowledge processing applications on-board vehicles may look up the KB to discover appropriate knowledge, given the needs and current context of driving of each vehicle. If well-formed input for a model is stored in the LDM, it can be applied locally if its bytecode is available on-board the ego vehicle. Figure 4.5 also indicates the possibility for vehicular nodes to exchange knowledge to and from the knowledge bases of other nodes. This aspect is discussed more in details in the next paragraphs.

A Structural Need for Knowledge Networking

As the bytecode and the semantic description of models is physically separated among vehicles and stored in different databases, a vehicle may hold a copy of a semantic description of a model without possessing a local copy of the corresponding bytecode. This allows for a flexible framework which supports the expansion or limitation of the creation and distribution of knowledge within a certain group of vehicles as needed.

In turn, a structural need appears for the exchange of bytecodes and semantic descriptions among vehicular nodes. Namely, requests should be defined for vehicles

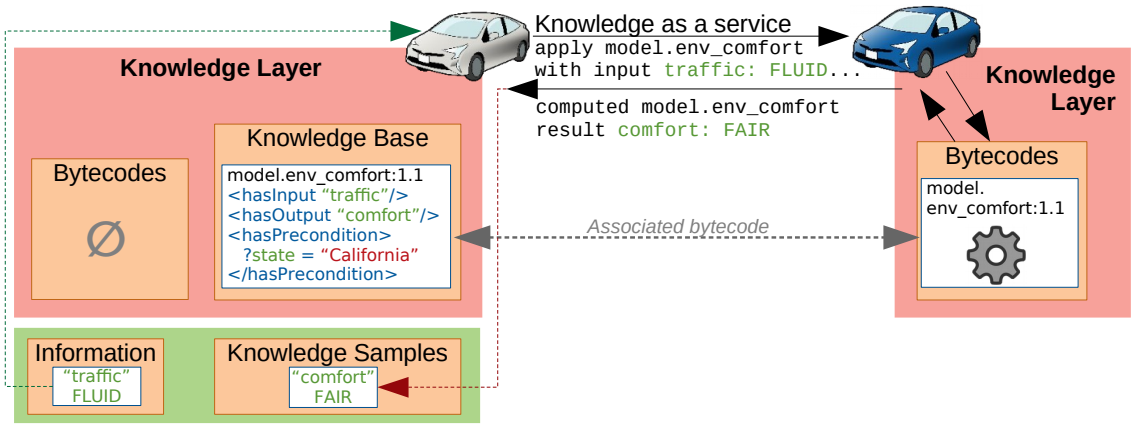


Figure 4.6: Semantic Requests for Knowledge Dissemination Between Two Vehicles

to fetch the bytecode which corresponds to a given model description, discover the description of models trained by other entities, or request the creation of knowledge samples from a remote vehicle which owns the corresponding bytecode, in a *knowledge as a service* operation.

As illustrated in Figure 4.5, the knowledge layers of vehicular nodes in a vehicular network may be interconnected. Namely, the KB is connected with the local bytecode storage of the ego vehicle, but may also request knowledge description and bytecodes from other nodes, to complete the local KB. The KB provides the basis for knowledge networking operations, as it lists the set of known knowledge models, their interface, and context of application.

Then, the KB is also responsible for knowledge synchronization between nodes. In addition to using unique names and version codes for each model, knowledge synchronization mechanisms should be defined to ensure that two remote nodes share a common understanding of a named piece of knowledge. Mechanisms of content synchronization have been implemented as part of ICN architectures, as surveyed in [163], and can be adapted for knowledge. Models could be divided between (i) ready-trained models, whose synchronization should follow described mechanisms, and (ii) models being trained, where no fixed version of the model is distributed. In that case, synchronization could take the age of the last contribution to the model into account.

4.1.4 Knowledge Dissemination

Provided that the semantic description and the bytecode of models is uncoupled, some nodes might own a copy of the bytecode of a model, whereas other only possess a copy of its semantic description. In turn, a knowledge dissemination messaging system should be defined to let vehicular nodes exchange knowledge from and to their knowledge layers, with other nodes.

For example, Figure 4.6 illustrates a case where two vehicles exchange knowledge about the `model.env_comfort` model, which we defined in Section 4.1.2. It shows the interaction between the knowledge layers of two vehicles, including the three

elements which we defined, i.e., (i) knowledge model descriptions, (ii) knowledge model bytecodes and (iii) knowledge samples, as well as information. Namely, the light gray vehicle on the right side of the figure owns a copy of the semantic description of the `model.env_comfort` model, as described in Figure 4.4, in its KB, but not the matching bytecode. On the other hand, the remote dark blue vehicle has a local copy of the bytecode.

In turn, the light gray vehicle could send a request to the dark blue vehicle to fetch a copy of the bytecode. Alternatively, it may request the creation of the knowledge directly by the dark blue vehicle, which is exemplified by Figure 4.6, which illustrates the a*knowledge as a service* workflow, where:

1. The light gray vehicle sends a request to create `model.env_comfort` knowledge to the dark blue vehicle, with a copy of the input information to use. To simplify, the figure only includes the traffic level input.
2. The dark blue vehicle receives the request, looks up its bytecodes list, finds the right bytecode and uses it to produce an output knowledge sample.
3. The output knowledge sample is returned to the light gray vehicle which stores it in its LDM.

The Impact of Semantics

In Figure 4.6, the light gray vehicle is sending a request to the dark blue vehicle to request the application of the named model `model.env_comfort` as a service. This can be done as the light gray vehicle already knows the semantic description of the model, hence it knows its full name and version code. Nonetheless, in the general case, vehicular nodes should be able to discover unknown knowledge which is owned by other nodes, i.e., knowledge whose name is unknown. As such, as analyzed in Section 3.1, nodes should be able to discover the name of an unknown model, based on a partial description of its semantic characteristics.

In turn, as part of the knowledge dissemination aspect of VKN, we contribute a set of messages which can be exchanged among vehicular networks to:

- Discover the name and the full semantic description of unknown knowledge possessed by other nodes, based on a partial description of its input, output, or context of use.
- Execute knowledge operations as described in Section 3.1, based on the discovered semantic description of models, such as request a copy of a bytecode, or request the creation of knowledge as a service to remote nodes.

In turn, the discovery and exchange of new knowledge as it is created by vehicles is allowed by the framework. For example, let us consider the case in which the light gray vehicle of Figure 4.6 is not aware of the existence of `model.env_comfort`. It only knows that it needs to assess a value of passenger comfort for a smart application and that it is currently driving in California. In turn, the vehicle can issue a request

to discover a model which match this partial description, e.g., to a centralized node owning an extensive KB, or through V2V communications. For example, it could issue a request stating: 'Please provide me the full semantic description and name of any model which can be applied in the context `?state = "California"` and which produces at least an output of type `Comfort`'. Upon reception of this request, the dark blue vehicle of Figure 4.6 can lookup its KB and return the full semantic description of `model.env_comfort:1.1`, which matches the request.

Supported Operations

To allow knowledge dissemination through VKN in vehicular networks, we define a set of messages to perform the following operations, as identified in Section 3.2: (i) Knowledge discovery, (ii) knowledge model bytecode transmission, and (iii) knowledge as a service. Knowledge discovery requests are designed to let vehicular nodes discover the full semantic description, including the unique name and version code, of a model based on an incomplete description or a set of semantic conditions, as introduced in the last paragraph.

Once a vehicular node has received a full copy of the semantic description of a model, it knows the unique name of the model, and may request either (i) to receive a copy of the bytecode of that model, to use it locally, or (ii) for another remote vehicle to create knowledge with that model on behalf of it, using the *knowledge as a service* operation, as illustrated by Figure 4.6. In the following paragraphs, we describe knowledge exchange requests which address each of the three aforementioned operations.

Knowledge Discovery Requests We define knowledge discovery requests as part of VKN. As introduced in the last paragraphs, knowledge discovery requests are used for a node to discover the name of an unknown knowledge model based on its characteristics. Knowledge models may be discovered based on a part of (i) their input or output parameters, and/or (ii) their context of application, as implemented in details in Chapter 6.

The issued requests list the named input and output items which should be provided by the model, as well as conditions on its context of application. An example pseudocode request is as follows:

- ```
  REQUEST MODEL DESCRIPTION
  INPUT #Road.Traffic, #Road.Visibility
  OUTPUT #Road.Comfort
  CONTEXT #State = "California"
  ```

The request is routed in vehicular networks using the underlying networking protocols which VKN is implemented over. In Section 6.2.2, we describe an implementation of VKN over NDN. In that case, knowledge exchange requests are routed using NDN interest routing protocols. In turn, a pseudocode response is provided:
  ```

→ `MODEL DESCRIPTION MATCH`
`NAME model.env_comfort:1.1`
`DESCRIPTION [OWL-S description]`

For an example of the OWL-S description of *model.env_ comfort:1.1*, see Figure 4.4.

Knowledge Bytecode Transmission Once vehicular nodes have obtained the name of a knowledge model, they may request a copy of the associated bytecode to a node which owns it. The following is a pseudocode example of a request and response for the *model.env_ comfort:1.1* bytecode.

- `REQUEST MODEL BYTECODE`
`NAME model.env_comfort:1.1`

→ `MODEL BYTECODE MATCH`
`NAME model.env_comfort:1.1`
`BYTECODE [machine code]`

Knowledge as a Service When a vehicular node is in possession of the name and the full semantic description of a model, as an alternative to requesting a copy of the bytecode of the model for a local application, a *knowledge as a service* operation can be requested, as analyzed in Section 3.2.

In turn, as shown with the light gray vehicle in Figure 4.6, a request may be issued for a remote node to create the knowledge on behalf of a requesting node, provided the well formed input which should be applied to the model. Below are a pseudocode request and its response for the remote creation of `model.env_comfort` knowledge as a service.

- `APPLY model.env_comfort:1.1`
`WITH INPUT traffic: FLUID, visibility: CLEAR, twoWheelers: LOW`

→ `COMPUTED model.end_comfort:1.1`
`BY [node_address]`
`RESULT comfort: GOOD`

What is more, nodes may wish to obtain the output of a known model applied with input collected in a distant context, i.e., without providing input values for the model application, but requesting both their sensing and their application as input to the model in a single request. Below is an example request of a vehicle wishing to obtain a value of the comfort level in a remote area. In that case, a remote vehicle which possesses a copy of the `model.env_comfort` bytecode in the target area: (i) Senses the required input information in that area, (ii) feeds them as input to the `model.env_comfort` model and (iii) returns the result to the requesting node.

- `APPLY model.env_comfort:1.1`
`IN [location]`

```
→ COMPUTED model.env_comfort:1.1
  IN [location] BY [node_address]
  RESULT comfort: FAIR
```

Cooperative Knowledge Training

In Section 4.1, we have contributed the definition of a VKN framework for the description, storage, and dissemination of fully trained knowledge models in vehicular networks. We did not explicitly cover the case of cooperative knowledge creation. Nonetheless, we believe the VKN framework to be adapted for the networking of knowledge being trained, provided appropriate semantics are defined. Namely, in addition to the VKN semantics defined in Section 4.1.2, semantics should be defined to describe (i) the training algorithm which is used, (ii) its parameters, e.g., the shape of an ANN, and the loss function to be used. The potential of this aspect of VKN is investigated in more detail in Section 4.3.

4.1.5 Discussion

In this section, we described a contribution as the definition of the VKN framework, divided in the aspects of knowledge description, storage, and dissemination. Through the combination of these aspects, VKN lets vehicles discover, store, and cooperatively use knowledge. A key contribution and choice is the physical separation of the bytecode of knowledge models and their semantic description. In turn, knowledge dissemination requests let nodes discover the description of unknown models and request either a bytecode copy, or their remote application through *knowledge as a service*.

After having described the architecture of the VKN framework and its key aspects, we perform two complementary studies to illustrate the impact and potential of the framework and to show preliminary performance improvements in complementary aspects:

- As a first step, in Section 4.2, we illustrate the impact of knowledge networking as supported by VKN on network efficiency metrics. Namely, we consider the overhead reduction of using VKN *knowledge as a service*, compared to the systematic transmission of complete model bytecodes.

- Then, in Section 4.3, we consider the opposite problem, i.e., the impact of knowledge networking on the accuracy of knowledge models themselves. Namely, we consider a use case of cooperative knowledge training through FL, where the training node selection is orchestrated by VKN and its context and model description semantics.

4.2 Evaluation of VKN-supported Knowledge as a Service

VKN is a conceptual framework to allow mutual understanding of knowledge models, which in turn provides a foundation to support knowledge-related operations. In this section, we perform a preliminary evaluation of the impact and potential benefits linked to VKN-supported *knowledge as a service* operation, as opposed to the traditional transmission of the bytecode of knowledge models, in terms of overhead. Namely, we run a simulation of (i) simplified vehicular mobility, and (ii) knowledge dissemination of comfort knowledge using the `model.env_comfort` model defined in Section 4.1. In turn, we compare the network overhead of dissemination using (i) the transfer of the model bytecodes with (ii) a VKN-supported *knowledge as a service* approach, where comfort knowledge is created in remote nodes on behalf of others.

While we use `model.env_comfort` as an example model, the evaluation setup does not implement actual knowledge creation mechanisms. Rather, it is model-agnostic in that it estimates and compares the overhead of knowledge creation for both the *knowledge as a service* and the *model bytecode transfer* approaches by counting and classifying the amount of transfers that would be required to satisfy all knowledge requests of all vehicles.

The considered scenario is described as part of three sections matching the (i) knowledge description, (ii) knowledge storage, and (iii) knowledge dissemination aspects of the VKN framework which we defined in Section 4.1. In this section, we contribute:

- A preliminary study to show the potential and impact of VKN on a networking performance metric, i.e., overhead.
- We show that the creation of knowledge through *knowledge as a service*, i.e., by a remote node on behalf of a requesting node, is beneficial in terms of overhead compared with the transmission of bytecode and input information to a requesting node for local knowledge creation, starting from a low threshold in the size of the considered model bytecode.

4.2.1 Knowledge Description Aspect

In this study, we reuse the `model.env_comfort` which we defined as an example in Section 4.1. It takes discrete qualifications of (i) the traffic conditions, (ii) the level of visibility, and (iii) the concentration of two-wheelers around a target vehicle to output a discrete value of passenger comfort on-board that vehicle, in the range `[GOOD,FAIR,POOR]`. A semantic description of the model using the OWL-S service description language, which we use as part of VKN, was provided in Figure 4.4. Then, a pseudocode simplified implementation was provided in Algorithm 1 as an example. Yet, the VKN framework is general and supports models implemented

using any AI-algorithm, such as ML, as long as the model has a set of at least one input and output.

A set of vehicles is simulated in a 1 square kilometer area divided into a grid of subareas, and follow a standard mobility model. Each vehicle periodically issues a request to obtain comfort knowledge about a subarea of the simulated space. In turn, the following types of messages, which have been defined as part of the VKN framework in Section 4.1.2, are exchanged among vehicles:

Model Bytecode 'Model Bytecode' messages carry the full bytecode of the considered model, i.e., the passenger comfort estimation model `model.env_comfort` in this case.

Input Information 'Input Information' messages carry the necessary input for the `model.env_comfort` model, i.e., (i) traffic, (ii) visibility, and (iii) concentration of two-wheelers. Typically, input information requests are sent by a vehicle in an area a to a vehicle in a distinct area b, to request for the sensing of the input information for the `model.env_comfort` model in area b. Input information messages are only used in the *model bytecode transfer* approach.

Model Discovery Request VKN 'Model Discovery' requests are initiated by a vehicle in an area a. They are broadcasted to neighboring vehicles in the same area a. If a vehicle in the area a owns a copy of the requested model in its KB, it becomes responsible for the knowledge application request. Model discovery requests are only used in the *knowledge as a service* approach.

Knowledge Samples VKN 'Knowledge Sample' messages are sent by a vehicle w which (i) owns a copy of the comfort-level model in its KB, and (ii) has received a knowledge creation service request from a distant vehicle v. They contain a value of `Road.ComfortLevel` which was produced by the local application of the model by w. Knowledge sample messages are only used in the *knowledge as a service* approach.

4.2.2 Knowledge Storage Aspect

Each simulated vehicle is equipped with a VKN knowledge layer as described in Section 4.1.3. What is more, each vehicle owns a copy of the semantic description of the `model.env_comfort` model in its KB, like the light gray vehicle in Figure 4.6. Yet, not all vehicles own a copy of the bytecode of the model. In turn, vehicles which want to obtain passenger comfort knowledge as the output of the `model.env_comfort` must use the knowledge dissemination aspect of VKN.

When a vehicle receives a copy of the bytecode of `model.env_comfort`, it keeps a copy in its local bytecode storage for the rest of the simulation, and will not need to request it from a remote node. In this study, the storage aspect of VKN is simple as the evaluation focuses on the network overhead related to knowledge dissemination, as described in Section 4.2.3.

4.2.3 Knowledge Dissemination Aspect

As introduced in Section 4.2.1, each vehicle moves in a square simulated area divided into a grid of subareas. Each vehicle periodically wishes to obtain the current value of passenger comfort in a specific subarea, i.e., cell of the grid. The frequency of 'knowledge retrieval requests' is detailed in the next simulation setup section. In this study, we assume a complete 1-hop V2V connectivity for the direct transmission of the knowledge exchange messages defined in Section 4.2.1.

The way of addressing knowledge requests of vehicles depends on the used approach: *knowledge as a service* or *model bytecode transfer*. As introduced in Figure 4.7:

- In the 'traditional approach' on the left side of the figure, i.e., model bytecode transfer, a vehicle v which wishes to obtain passenger comfort knowledge from a remote area a will first obtain a copy of `model.env_comfort`, if not already cached in its KB, and input sensed from a, if the vehicle v is not itself driving in a. In turn, the knowledge will be locally created by v. As such, the transfer of input information and of model bytecode copies is uncoupled. The following scenarios are possible:
  - The vehicle owns no copy of the `model.env_comfort` bytecode, nor input information (it does not drive in the target area hence cannot sense it itself).
  - The vehicle owns a copy of the bytecode in local storage, but no input information from the target area.
  - The vehicle owns input information, as it is driving in the target area, but does not own a copy of the model bytecode.
  - If the vehicle has got both input information, i.e, it drives in the target area, and a copy of the model bytecode, no communication is required to create the knowledge sample.
- In the 'remote knowledge creation' approach on the right side of the figure, i.e., knowledge as a service, v directly requests the application of the model to a remote vehicle which has the bytecode in its KB.

In turn, Figure 4.8 is a flow chart which illustrates the detailed process of knowledge retrieval when a vehicle v requests knowledge for both the *knowledge as a service* approach in red solid lines, and the systematic *model bytecode transfer* approach in blue dashed lines.

What is more, Figure 4.9 summarizes the VKN-supported architecture of the *knowledge as a service* creation of knowledge from the `model.env_comfort` model. As introduced in Section 4.1.4, the `ego_car` vehicle on the left side of the figure produces *knowledge as a service* queries for the application of the model in a specific context, i.e., in this case, in a specific area. The requests are sent to a vehicle `remote_car` in the target area which owns a copy of the model in its KB, with the requested version code `1.1`. In turn, `remote_car` senses the required input for

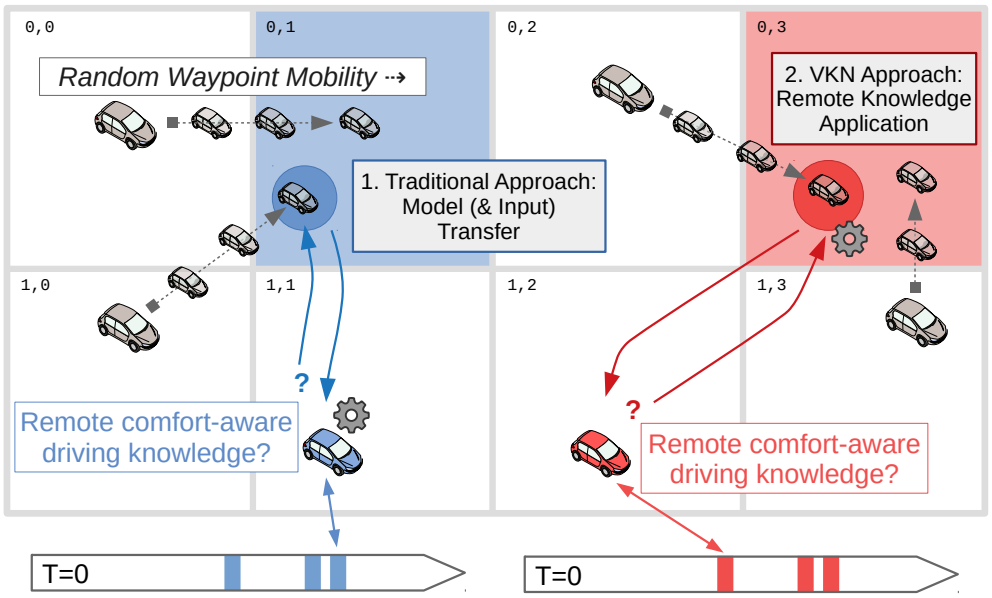


Figure 4.7: Knowledge Distribution Overhead Evaluation Setup

the `model.env_comfort` model, locally produces the comfort level output *knowledge sample*, and returns it to `ego_car`.

4.2.4 Simulation Setup, Results & Discussion

In this section, we summarize the simulation setup and provide its detailed parameters. In turn, we present the results which compare the overhead of the VKN *knowledge as a service* approach with that of a systematic *model bytecode transfer* approach, and discuss the impact of the obtained findings.

Simulation Setup

The parameters of the simulation setup presented in this subsection are summarized in Table 4.1. A set of vehicles is simulated in a $1km^2$ area, which is divided into a square grid. Each cell of the grid represents a distinct area in which input information for the `model.env_comfort` can be sensed, i.e., the traffic conditions, visibility, and concentration of two-wheelers in the area.

Mobility In this study, we aim at showing the potential and impact of VKN *knowledge as a service* operations. It is a preliminary work to illustrate an initial indication of the need for VKN-supported knowledge operations. As such, rather than a realistic mobility pattern, as well as a realistic model, we use random mobility models which are statistically controllable. Despite not realistic, they allow to show a pattern and illustrate the potential impact of VKN. After showing the initial impact

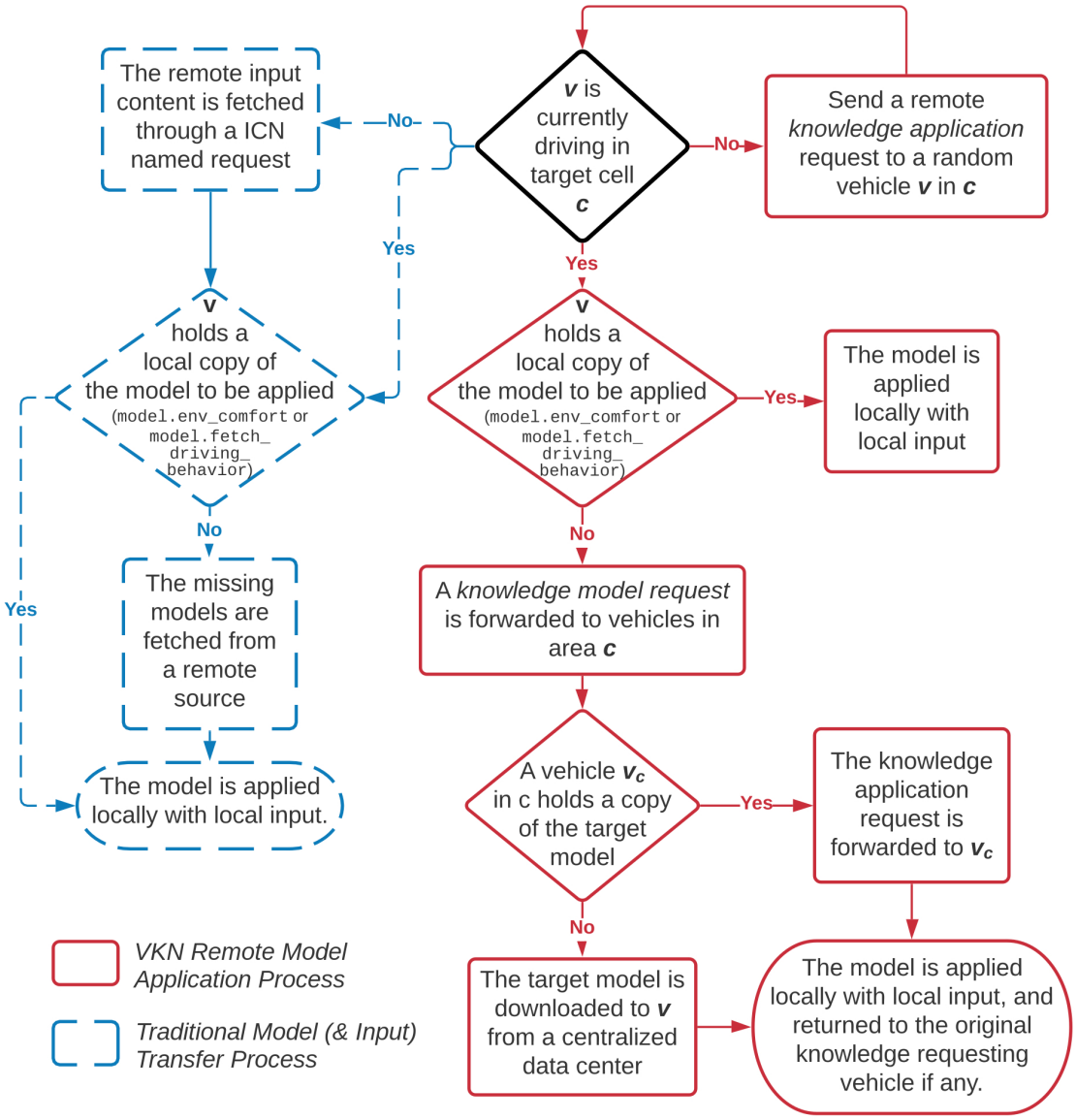


Figure 4.8: Comparison of the 'Knowledge as a Service' and 'Model Transfer' Processes

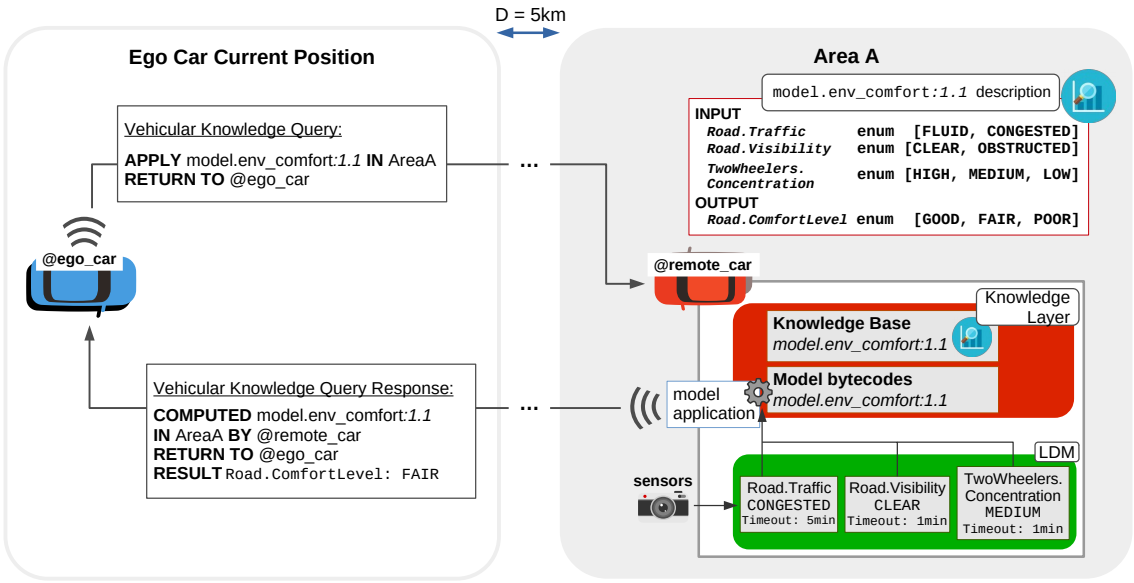


Figure 4.9: Comfort Level Retrieval in a Remote Area using VKN-supported Knowledge as a Service

Table 4.1: Simulation Parameters for the Evaluation of Knowledge as a Service Overhead

Parameter	**Value**
Surface of the simulated area	$1km^2$
RWP mobility speed range	$s \in [2, 20]m/s$
RWP pause probability at each stop	$P_{pause} = 0.5$
RWP maximum pause duration	5min
Poisson Frequency of knowledge requests	$\lambda = \frac{1}{30}s^{-1}$
Number of vehicles (urban scenario)	1000
Number of cells (urban scenario)	$20x20$ ($50m^2$ cells)
Number of vehicles (rural scenario)	200
Number of cells (rural scenario)	$5x5$ ($200m^2$ cells)
Evaluation Metrics	
Amount of transmission per message type for the *knowledge as a service* setup.	
Amount of transmission per message type for the systematic *model bytecode transfer* setup.	

and potential benefits, Chapters 5 and 6 investigate VKN using realistic models and mobility.

In turn, in this preliminary study, vehicles' movements are simulated following a steady-state Random WayPoint (RWP) mobility model [164]. Initially, each vehicle either pauses with a probability of $P_{pause} = 0.5$ for up to 5 minutes of simulated time or follows a straight line trajectory at a constant speed sampled from $s \in [2, 20] m/s$, between a start and a destination point. Upon reaching its destination, the vehicle either pauses (with P_{pause} probability, for maximum 5 minutes) or chooses a new destination point and speed. The process repeats until the simulation is ended.

Knowledge Requests A timeline is generated which contains R=10000 knowledge retrieval requests. Each vehicle in the simulation periodically sends a knowledge retrieval request targeting a specific area, i.e., a grid cell. The frequency of the requests of each vehicle follows a Poisson process with a 30 seconds rate. The way of addressing the knowledge requests of vehicles depends on the used approach: *knowledge as a service* or *model bytecode transfer*, as described in Section 4.2.3. The simulation ends when all requests have been answered.

Urban & Rural Scenario We run both simulation setups, i.e, the *knowledge as a service* and *model bytecode transfer* approaches, in two distinct scenarios which simulate, respectively, (i) urban and (ii) rural mobility. Namely, the 'urban' scenario aims to imitate urban conditions as it features a vehicle density of 1000 vehicles/km^2 in a grid divided into $50m^2$ cells. The 'rural' scenario features a vehicle density of 200 vehicles/km^2 in a grid of $200m^2$ cells.

100 experiments are run for both the 'urban' and the 'rural' scenarios. Each experiment is composed of one simulation using the *remote model application* approach, and another one using the traditional *model transfer* approach. As such, 400 simulations of vehicular mobility and knowledge retrieval are performed in total. Each simulation uses a distinct random seed to initialize the RWP mobility model. At the end of each simulation, the number of transmissions are counted. What is more, each transmission is associated with its type, i.e., (i) model bytecode, (ii) input information, (iii) model discovery request, and (iv) knowledge samples, as defined in Section 4.2.1.

Results

Table 4.2 illustrates the obtained results for both rural and urban scenarios, and for both the *knowledge as a service* and systematical *model bytecode transfer* approaches. The average number of transmissions for each message type are counted, as well as the associated 95% confidence intervals obtained from the 100 simulations which were run in each case. In the urban scenario, the amount of *model bytecode* transfers was reduced by $15 \pm 0.2\%$ using the VKN-supported *knowledge as a service* approach, compared with the systematical *model bytecode transfer* approach. In the rural case, the amount of model bytecode transfers was reduced by $44 \pm 0.4\%$.

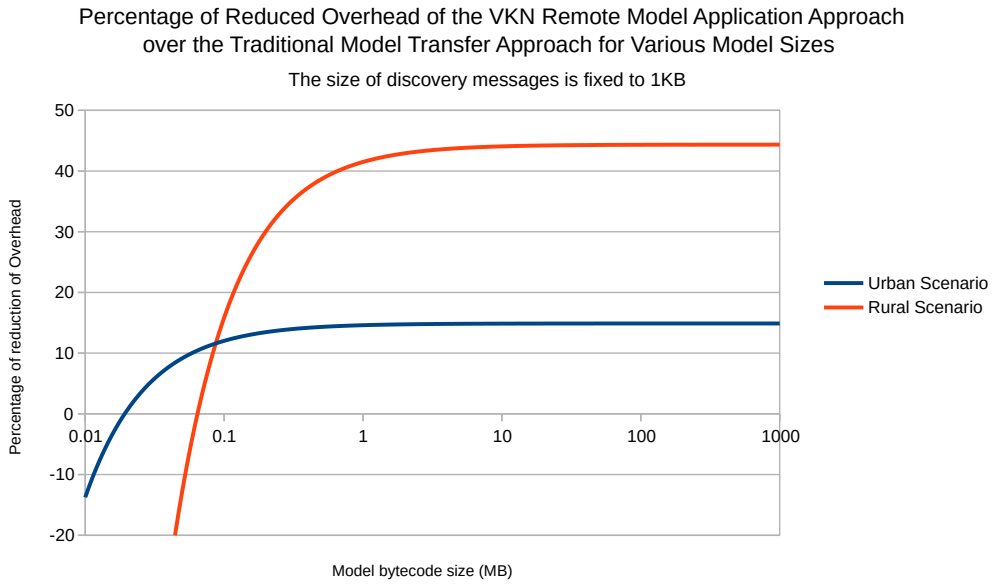


Figure 4.10: Overhead of VKN-supported Knowledge as a Service Compared with Traditional Static Model Transfer

The obtained estimation of the number of transmissions of each type of message can be used to estimate the overhead associated with the VKN *knowledge as a service* and the systematical *model bytecode transfer* approaches. In practice, the size of the bytecode of a knowledge model depends on its AI technology of implementation and complexity, and can range from megabytes to hundreds of megabytes for trained ANNs. In turn, depending on the considered model, the VKN-supported approach of *knowledge as a service* allows a moderate to strong reduction of overhead in terms of knowledge transfer, compared with the systematical *model bytecode transfer* approach. An analysis of the obtained results was performed in our publication in [18], as follows:

> While an extra 0.3 to 0.6 model discovery message[s] per request were transmitted using the VKN [knowledge as a service] approach for respectively the urban and rural scenarios, their size is negligible in front of the bandwidth saved through reduced model transfers. Considering an equivalent size of model input and output, the VKN [knowledge as a service] approach becomes beneficial in terms of overhead from a model size of 100 kilobytes. It reaches, from 1 megabyte, a stable 14 or 40% overhead reduction in terms of model transfers for respectively the urban and rural scenarios [as illustrated by Table 4.2 and Figure 4.10]. What is more, using [the] VKN [knowledge as a service approach], model input [information] transfers were traded with an equivalent amount of [knowledge sample] transfers for the urban scenario, and a slight increase of 2% for the rural scenario. Depending on the model, [if the produced output knowledge samples are] lighter than the input [they were] extracted from, overhead can be further reduced.

Table 4.2: Amount of Communications Per 10000 Requests in the Urban and Rural Scenarios

Transmissions per 10000 requests:	Urban scenario (1000 vehicles)		Rural scenario (200 vehicles)	
	Knowledge as a Service	*Model Transfer*	*Knowledge as a Service*	*Model Transfer*
Model Bytecode	851 ± 2	1000	111 ± 2	200
Knowledge Sample	9013 ± 11	0	9822 ± 3	0
Input Information	0	9006 ± 11	0	9592 ± 4
Model Discovery	2864 ± 12	0	5703 ± 30	0
Overhead Comparison:	Urban scenario (1000 vehicles)		Rural scenario (200 vehicles)	
	Knowledge as a Service	*Model Transfer*	*Knowledge as a Service*	*Model Transfer*
1MB / Bytecode	0.85GB	1GB	0.11GB	0.2GB
1KB / Discovery	2.9MB	0	5.7MB	0
Total overhead	0.86GB	1GB	0.12GB	0.2GB
Difference	14%		40%	

Discussion

In this section, a preliminary evaluation of VKN on a passenger comfort level knowledge retrieval use case was performed. The estimated overhead of knowledge retrieval through VKN-supported *knowledge as a service* operations, in which knowledge is created by a remote node on behalf of a requesting node, is compared with that of systematical *model bytecode transfer* operations in which the requesting node retrieves both the model bytecode and the necessary input for local knowledge creation. This preliminary evaluation is meant to show the potential and impact on a network performance metric of VKN-supported knowledge exchange operations, using a statistically controllable random mobility model, as well as a simple model case, even though the evaluation is model-agnostic. In turn, this evaluation shows both (i) the potential and impact of VKN on the network performance of knowledge dissemination, and (ii) the need for a realistic evaluation, both in terms of mobility and used knowledge model, to confirm the performance improvements related to VKN, which, in turn, is the topic of Chapters 5 and 6.

This study showed the potential and impact of VKN applications on a networking performance metric. In the next section, we perform a complementary preliminary study to evaluate the impact of networking, i.e, the dissemination of model descriptions and driving context of vehicles, on VKN applications, i.e., the accuracy and speed of cooperative knowledge training by vehicles.

4.3 Evaluation of VKN-supported Cooperative Model Training

In Section 4.2, we have demonstrated the potential impact of VKN on networking and preliminary performance gain through the creation of knowledge from fully trained models as a service, in terms of overhead. In this section, we evaluate

the potential and impact of networking on the performance of VKN applications, through the case of cooperative knowledge training by a set of vehicles. Namely, we implement and evaluate VKN *cooperative knowledge training* on a simplified case, through the VKN-supported orchestration of federated learning training processes, and show preliminary performance improvements.

Choice and Impact of FL FL is a body of algorithms which address the cooperative training of a ML model by multiple nodes. In this section, we focus on FL to perform the training of a supervised model, i.e., a model which is trained using a set of labeled training entries. A description and analysis of FL and cooperative knowledge model training approaches in more details was performed in Section 3.2. Namely, we find that applications of *cooperative model training* require two parallel building blocks to be functional:

1. The definition of cooperative model training algorithms, such as FL or GL, or alternative approaches such as DML, as introduced in Section 3.2.
2. The definition of orchestration mechanisms to select nodes which are able to take part in training, i.e., possess the right training data, and whose training data has been sensed in the right context for the model being trained.

In this work item, we investigate on the orchestration of FL through VKN, and the exchange of semantic context content among vehicles. As analyzed in Section 3.2, the vehicular environment has a highly dynamic topology, and is a challenging environment to let vehicles perform cooperative knowledge learning, as the driving context, and in turn the training environment, of each vehicle is highly dynamic and volatile. In turn, the orchestration of FL in an environment with such constraints requires a constant exchange and synchronization of training environment data among nodes. What is more, direct data exchange between vehicles is limited for privacy and liability reasons. In turn, FL is an appropriate technology as it is designed to function in such distributed environments and maintain data privacy. Yet, it features challenges related to the orchestration of training in the dynamically evolving vehicular environment. As such, we believe that the orchestration of FL training processes is a relevant use case to show the potential and impact of VKN.

This orchestration of the selection of training nodes requires the definition of training semantics, which allow a centralized coordinator or the nodes themselves to query other nodes about their training capacity and the context of collection of their training data. What is more, as described as part of the VKN framework definition in Section 4.1.4, semantics are required to describe the model being trained: (i) its interface, i.e., input and output content, (ii) the required context of the collection of the training data, and (iii) training-related parameters, such as the optimization algorithm or loss function.

It is in this aspect that the knowledge description semantics described for the VKN framework in Section 4.1 can be used to support the orchestration and functionality of *cooperative model training* operations in vehicular networks. In this section, we implement *cooperative model training* orchestration of a FL model training problem using the VKN framework. In two case studies taking different aspects

of vehicular mobility into account, we demonstrate accuracy improvements of FL model training processes in vehicular networks using a popular benchmark model for FL training. As such, this section does not contribute to FL or GL algorithms themselves, but uses VKN to orchestrate the training vehicle selection mechanisms to select vehicles which can contribute to the training in the right context, and reach performance improvements independently of the specific model being trained.

Plan & Contributions To begin with, we describe in more details the challenges related to the orchestration of the training nodes selection for cooperative training in the vehicular environment. Then, we describe the architecture of the VKN-supported cooperative training orchestration which we define, following the three main aspects of the framework: (i) knowledge description, (ii) storage, and (iii) dissemination. Then, we summarize the setup of a preliminary simulation to compare the VKN-orchestrated FL training to a traditional orchestration which does not take the context of training into account. Finally, we analyze and discuss the results and the potential impact of VKN for cooperative model training, before opening on the need for Chapter 5 and 6.

In this section, we contribute:

- The integration of the VKN framework with cooperative and distributed model training in vehicular networks.
- An algorithm to perform the selection of training nodes in a pool of vehicles, based on their last known training environment.
- We show performance improvements in terms of training speed and accuracy of a benchmark FL model when the training node selection is orchestrated by VKN, rather than a traditional context-agnostic approach.

4.3.1 Requirements for Vehicular FL Orchestration

As introduced in the previous section, apart from the training algorithms, the accuracy of cooperative model training processes in vehicular networks depends on the training orchestration mechanisms, i.e., the selection of the most relevant training nodes. The selection of appropriate training nodes faces a set of challenges which were addressed in other works, such as channel link quality or computational power [165, 166], security [153, 154], or the definition of incentives for nodes to contribute to the training [156, 157].

In this work, we focus on the specific challenges of FL related to the mobility of nodes in the vehicular environment, as introduced in [167]. Namely, the access of vehicles to training data and the relevance of the context in which the training data were gathered by vehicles should be considered when orchestrating vehicular *cooperative model training*. In turn, semantic description mechanisms to describe the required data type and context of training, as well as the data type and context of training on-board vehicles should be defined. Eventually, we integrate VKN semantics to address these requirements.

Vehicular Mobility Challenges for FL

As described in Section 3.2.2 and Table 3.1, existing approaches have been considered to optimize the selection of training nodes for FL training processes in mobile environments. They involve the following categories:

- Resource constraints-based approaches, e.g., computational power, channel link quality, or energy level.
- Training data distribution-based, i.e., to handle a non-IID distribution of data among nodes.
- Security-based, e.g., to avoid selecting untrusted training nodes.
- Miscellaneous approaches, such as incentive-based or gradient descent convergence optimizing approaches.

Yet, to the best of our knowledge, no approaches take the fast-evolving training context which is inherent to the vehicular environment into account. Namely, as a consequence of the high vehicular mobility, both (i) the nature of the training data sensed by vehicles and (ii) their context of sensing change dynamically over time. For example, let us consider the cooperative training of a model to assess the exit probability of vehicles in 4-entry roundabouts. In turn, being able to take part in the training of this model requires the vehicles to have crossed such a roundabout recently, such that the data maintenance mechanisms of its local storage did not erase the potential training data. What is more, if a vehicle drives through a 7-entry roundabout, it will possess the right data type for training, i.e., the kinematics of a vehicle sensed in a roundabout. Yet, the data has not been obtained in the right *context*, i.e., 4-entry roundabouts.

In turn, in these conditions, it is critical to determine which nodes are able to train a model, and for how long, as they are currently featuring the right training context. As described in Section 3.2.2, this aspect is not considered in current FL orchestration approaches, which either perform studies where the training context of each vehicle is preset to match the requirements, or is known in advance.

As such, the fast evolving training environment for each vehicle, due to the high mobility, is a broad problematic, which emphasizes the need for a VKN orchestration of FL training node selection processes. In turn, we exemplify this broad problematic of training context-aware FL orchestration through two complementary case studies of vehicular *cooperative model training*. We quote the description of each case study from our paper published for this work item in [19].

Case Study 1: Commonly Observable Input

> We consider the use case of a FL model that can be trained from input that can be observed by vehicles at all times while driving. Vehicles sense training input as they drive using on board sensors and are then connected with a FL coordinator to train a model from their locally

observed data. In that case, the trained model uses input data that can be collected at any time of driving by vehicles, e.g., road-state related information.

For example, [58] describes a machine learning model of road roughness estimation from vehicles' front camera. Although the authors train the model using traditional machine learning, the model can be adapted for FL training, where sensed training information is kept on board vehicles.

At each step of FL training, the coordinator performs client selection and chooses a set of vehicles to locally compute a model update. The nature of mobility in vehicular networks brings additional challenges to perform efficient FL client selection. [As illustrated by Figure 4.11,] mobility in vehicular networks is nonrandom. Two originally distant vehicles are likely to drive the same roads at some point in their journey, as they drive through main arteries, e.g., in cities.

As such, the unequal distribution of vehicles on the road network leads to many vehicles sensing similar training input in similar areas, e.g., camera images of main roads. The data distribution is scattered, as identified as a challenge of FL in vehicular environments in [167]. A mechanism must be designed to unsure that nodes possessing diverse training data sensed in secondary roads also get represented in FL training, to avoid overfitting the model to typical main axes mobility.

Case Study 2: Sparsely Observable Input

We consider the use case of a FL model that can be trained from input that can only be observed by vehicles at punctual times and locations. For example, [59] presents a machine learning model to predict near-crash situations from observed vehicles' kinematics. Near-crashes are reasonably rare events that can only be observed at punctual points of space and time. In the hypothesis of training this model using FL, only vehicles having sensed kinematic data of another vehicle experiencing a near-crash should be leveraged as local training nodes. As the vehicles are moving, the ability of vehicles to take part in training dynamically evolves through time.

In order to accurately train the model, the selected clients should be nodes having witnessed a near-crash event and sensed the accompanying kinematics information. [As illustrated by Figure 4.12,] an orchestration mechanism must be defined in order to allow the coordinator and training vehicles to communicate training capacities and allocate training resources to vehicles that hold meaningful input.

As such, our contributions in terms of training context-aware orchestration of FL training are twofold:

- *Case study 1* represents the case where training data is available on board vehicles and sensed in the right context, but a specific location of data sensing

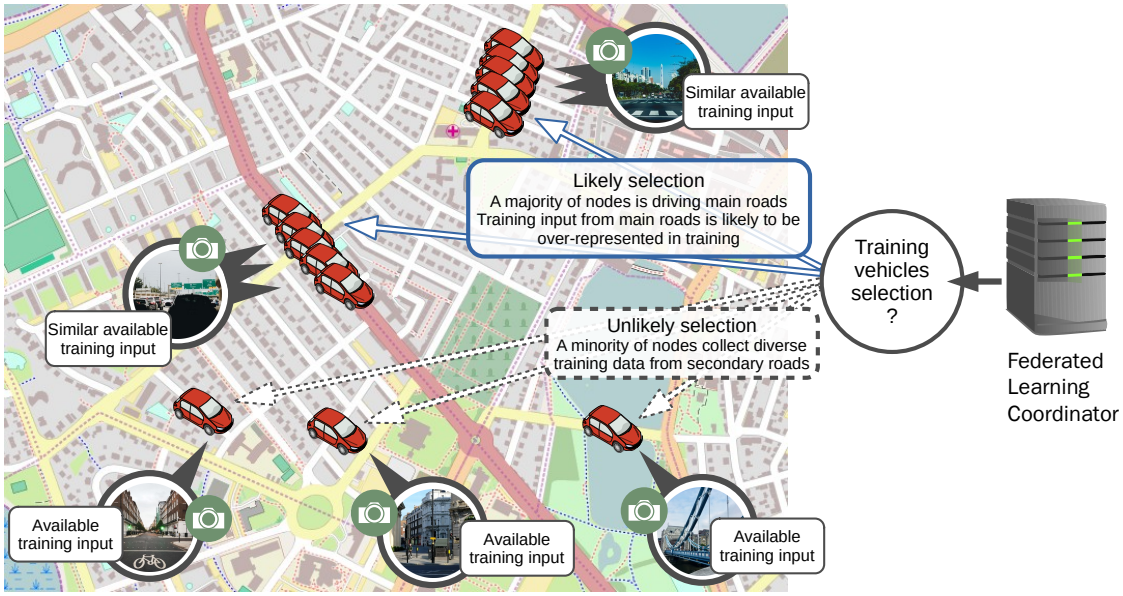


Figure 4.11: Vehicular Mobility Impact for FL with Commonly Observable Data

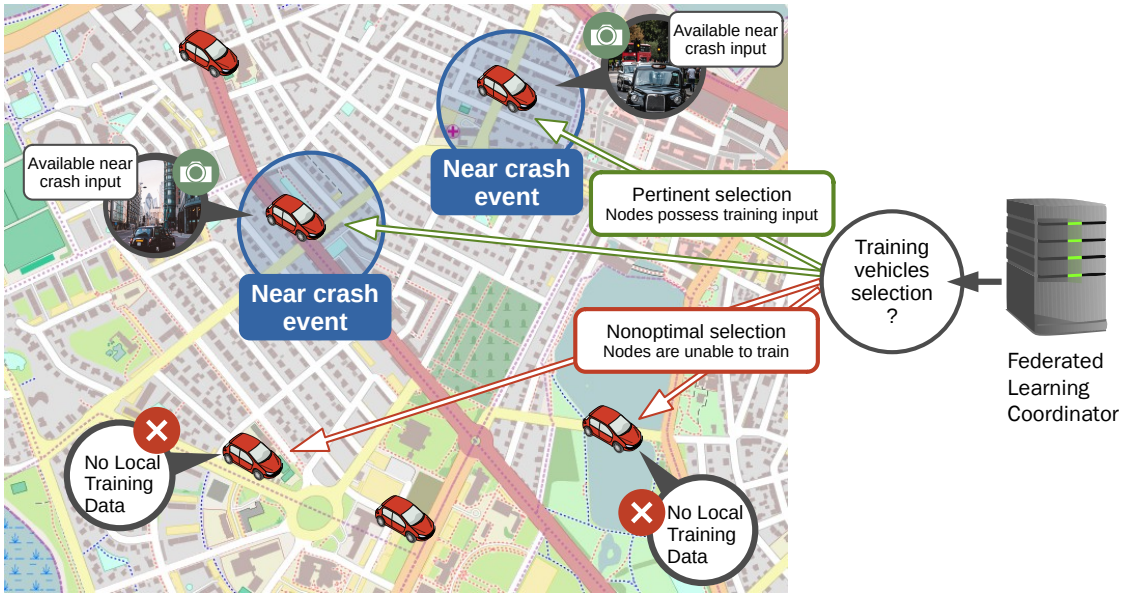


Figure 4.12: Vehicular Mobility Impact for FL with Sparsely Observable Data

is over-represented. For example, to train a model assessing the exit probability of vehicles in 4-entry roundabouts, data collected from major 4-entry roundabouts in primary roads is likely to be over-represented compared to data sensed in secondary 4-entry roundabouts, which must be taken into account. As such, in this work item, we contribute a training vehicle selection orchestration mechanism which ensures the fair representation of training data collected in various locations featuring the right training contexts, which we name **input variety orchestration**.

- *Case study 2* represents the case where training data sensed in the right context is only available on board a limited number of vehicles. For example, only vehicles which have witnessed a vehicle exit a 4-entry roundabout in recent history. Then, only those vehicles which feature the right nature of training data sensed in the right context must be selected for training. In turn, in this work item, we contribute a training vehicle selection orchestration mechanisms which increases the likelihood that the selected vehicles feature the right type and context of training data, which we name **training context availability orchestration**.

Contributions & VKN Framework Integration

The **input variety orchestration** and **training context availability orchestration** are two key aspects which must be addressed to orchestrate FL tasks in real vehicular environments where the training environment of vehicles evolves dynamically. Respectively, (i) a selection of training vehicles which maximizes the diversity of training data, while (ii) ensuring it has been sensed in the right context, is needed.

In this section, we contribute protocols to allow training vehicles and potential centralized coordinators to communicate about the required training environment to train a specific model, e.g., the input/output interface, the context of training input sensing, and training parameters of the model to be trained.

Using the VKN framework which we described in Section 4.1, we define all four of:

- A semantic format for vehicular nodes to communicate about their training environment, i.e., the named data types which they possess in their local storage and their context of sensing.
- A semantic format for vehicular nodes such as a centralized FL training coordinators to describe the required training environment of a model, to match it with the training environment of potential training nodes.
- An algorithm of orchestration of the selection of FL training vehicles, which contributes both the **input variety orchestration** and **training context availability orchestration** concepts.
- A simulation of the FL training of a benchmark model by a pool of vehicles,

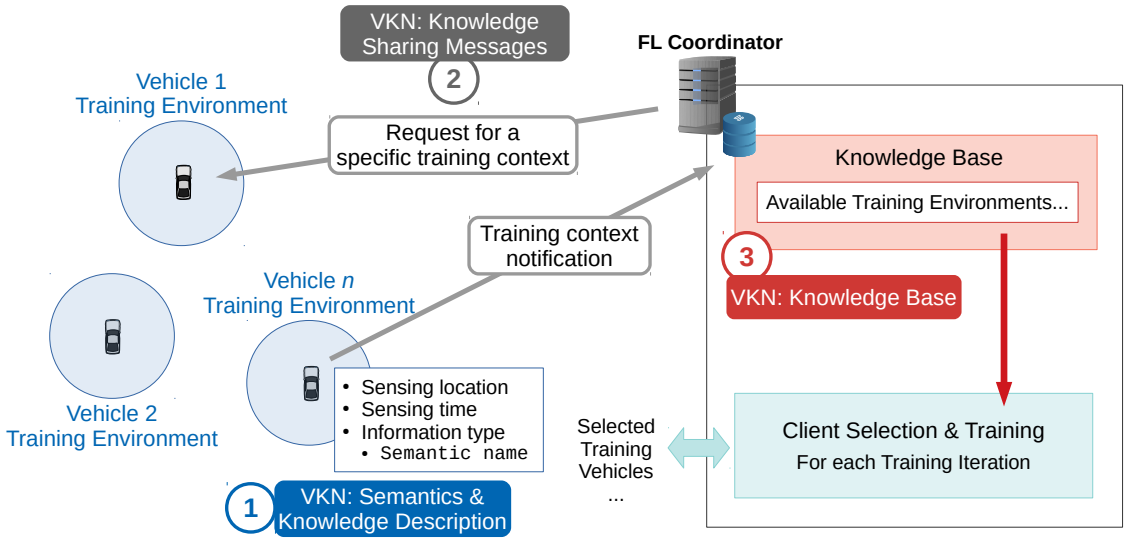


Figure 4.13: FL Orchestration Through VKN [19]

in which we compare a traditional FL training node selection mechanism with the aforementioned orchestration algorithm.

In the next sections, we describe the VKN-supported orchestration of FL training by vehicles through the three core aspects of the framework, i.e., (i) knowledge description, (ii) storage, and (iii) distribution. Figure 4.13 illustrates an overview of the integration of *cooperative model training* with VKN, describing the role of each aspect, which we comment, respectively, in Sections 4.3.2, 4.3.3, and 4.3.4.

4.3.2 Knowledge Description Aspect

The following technical requirements should be defined to allow mobility and training environment-based orchestration of *cooperative model training* in vehicular networks, as quoted from our publication in [19]:

"1. A message format should be defined to describe the nature of information sensed by vehicles, and the context it is sensed in. This message should include at least:

   - A standard, semantically-defined name to identify the nature of sensed information.
   - The time and position of information collection.

   Additional pertinent contextual information includes (i) the status of the vehicle: driving, idle, (ii) data on vehicle kinematics and destination.

2. A message format should be defined to formally describe FL models and their required input. Through a formal definition of the conditions of application of a model, coordinators can perform efficient training node selection by targeting the nodes that possess the required training input in the right format.

```
model_name: model.road_surface_condition

model_description:
    INPUT(camera_image: #Image.FrontCamera),
    OUTPUT(road_surface_condition: #RSC.ClassID)

model_training_parameters:
    MODEL_PARAMETERS(... [Neural Network
                      Weights Description]),
    LOSS_FUNCTION(loss.cross_entropy)
```

Figure 4.14: Example of a VKN Model Description, based on [58] (Case Study 1)

Elements to formally describe a model should at least include input and output information types and nature.

3. A message format should be defined for coordinators to request for vehicles that possess the right context for training a model."

The knowledge description part of VKN which uses semantics to describe the interface, i.e., input and output information types, as well as the required context of application or training of a model, can be used to describe the requirements in terms of training data on-board vehicles to train the model. Using such model description semantics, the vehicles and the potential coordinator are able to reach a common understanding on the input requirements for models to be trained cooperatively.

Figure 4.14 illustrates a pseudocode example of VKN knowledge description for a model which estimates the condition of the road surface, as described in [58] and introduced in *Case Study 1*. The model takes a camera image as input and outputs an indicator of road surface condition. In addition, a description of the training parameters for the model should be provided in the VKN description of the model, as illustrated in the `model_training_parameters` section of Figure 4.14. The training parameters include the algorithm used to train the model, e.g., ANN, and additional parameters such as the loss function used to assess the quality of the model. Depending on the training algorithm, specific parameters should also be included, such as a description of the shape of the ANN, in the case of a model implemented using neural networks.

4.3.3 Knowledge Storage Aspect

As indicated in point 2 of Figure 4.13, at the beginning of the model training, the coordinator sends a request of training environment to a set of potential training vehicles. In parallel, as the vehicles move, they notify the coordinator dynamically of their training environment update, i.e., whether they are able to train the requested model.

The coordinator maintains a local KB keeping track of the list of vehicles whose training environments match the required environment to train the considered model.

What is more, the reception time of the last training environment notification from each vehicle is saved, as well as the location of the vehicle at the sensing time of its last notification.

In turn, at each iteration of model training, the coordinator is able to make an educated decision on which training vehicles to select for local training based on the training environment knowledge available in its KB. The point 3 of Figure 4.13 illustrates this aspect. The approach can be adapted to a decentralized GL context by letting each vehicle maintain a KB containing the training environment of their neighboring vehicles.

Namely, to select the training vehicles at each iteration, the coordinator can:

- Extract a list of vehicles which are likely to feature the right types of training data for the model from its KB.
- Select a set of training vehicles from this list, based on the context of training data sensing faced by the vehicle during its last training environment notification. In this example, the context of sensing of the training data is defined by the location and the time of sensing of the data.

4.3.4 Knowledge Dissemination Aspect

As illustrated by points 1 and 2 of Figure 4.13, provided with a description of the interface of the model to be trained, the vehicles and the potential coordinator must be able to communicate so that:

1. The coordinator can request for vehicles which feature the right training environment, i.e., possess data which matches the required input data type for the considered model, e.g., `Image.FrontCamera` data.
2. The vehicles can answer by communicating their training environment to the coordinator, or to each other in case of decentralized GL training.

VKN knowledge exchange messages as introduced in Section 4.1 can be used to support the communication of training environments between vehicles and training coordinator. Figure 4.15 provides a pseudocode example of requests and answers which can be communicated between a training coordinator and a vehicle, in order to exchange training environment knowledge. The description of a training environment includes whether the requested data type is available on board the target vehicle. If it is available, the time and location of sensing of the data is also included.

In turn, we contribute a VKN-based, training environment-aware FL training vehicle selection algorithm, which is listed in Algorithm 2. It integrates both the **input variety orchestration** and **training context availability** contributions. To summarize, the VKN-orchestrated approach takes the knowledge of the training environment of each vehicle into account in two complementary ways:

Algorithm 2 VKN-supported Training Environment-Based FL Training Vehicle Selection Mechanism

Variables

$KB = key\_value\_list()$
$select\_size = 10$
$min\_dist = 50m$

procedure RECEIVE_TRAINING_ENVIRONMENT($vehicle$, $data\_type$, $location$, $time$)
    $KB[vehicle] \leftarrow (data\_type, location, time)$
end procedure

procedure MAINTAIN_KB($current\_time$)
    for each $(vehicle, data\_type, location, time)$ **in** KB **do**
        if $current\_tme - time > 5min$ **then**
            $KB.drop(vehicle)$
        end if
    end for
end procedure

procedure TRAINING_VEHICLE_SELECTION($current\_time, optimize$)
    $maintain\_KB(current\_time)$
    $selected\_vehicles = list()$
    ▷ Not enough known training environments to select the requested number of training vehicles.
    if $KB.size() < select\_size$ **then**
        $selected\_vehicles \leftarrow KB.keys()$
        for $i = 0$ **to** $(select\_size - KB.size())$ **do**
            $selected\_vehicles.add(random\_vehicle())$
        end for
    else
    ▷ Produce $select\_size$ clusters based on the *location* of training environments in the KB.
        $clusters \leftarrow KMeans(KB, select\_size)$
        for $i = 0$ **to** $clusters.size()$ **do**
            ▷ Add the vehicle with the most recent known training environment of the cluster.
            $entries.sort\_by\_desc('time')$
            $selected\_vehicles.add(entries[0]['vehicle'])$
        end for
    ▷ Optimization: Remove vehicles whose training environment locations are too close to each other.
        if $optimize$ **then**
            for $i = 0$ **to** $selected\_vehicles.size()$ **do**
                $vehi \leftarrow selected\_vehicles[i]$
                for $j = i + 1$ **to** $selected\_vehicles.size()$ **do**
                    $vehj \leftarrow selected\_vehicles[j]$
                    if $distance(KB[vehi]['location'], KB[vehj]['location']) < min\_dist$ **then**
                        $selected\_vehicles.drop(vehj)$
                        break
                    end if
                end for
            end for
        end if
    end if
return $selected\_vehicles$
end procedure

```
Coordinator requests:
 REQUEST TRAINING_ENV(#Image.FrontCamera)
         IN [Area] AT [Time Range]

Vehicle answers:
 AVAILABLE TRAINING_ENV(#Image.FrontCamera)
           IN [Area] AT [Time Range],
     VEHICLE ID: [vehicle address]
```

Figure 4.15: Training Environment Request and Answer, based on [58] (Case Study 1)

1. To begin with, the vehicles selected for training at each step are extracted from the list of vehicles which have communicated their relevant training environment to the coordinator. If less than 10 vehicles have communicated their relevant training environments, additional random training vehicles are chosen until 10 vehicles have been selected.

2. What is more, if more than 10 vehicles have communicated their relevant training environments, the VKN-orchestrating coordinator performs an optimization of the training vehicle selection based on the known context associated with the training environment of each vehicle. In this case the context of the training environment of a vehicle is defined by the last location of sensing of the relevant training input.

   As such, the training vehicle selection is optimized such that the selected vehicles possess diverse contexts of training input sensing, in order to represent training data from various contexts and not over-represent one context of data sensing. The optimization is implemented as follows:

   - The coordinator extracts from its KB all known training environments which have been declared to be relevant for training by vehicles. The obtained knowledge takes the form of a list of entries featuring (i) the address of the associated vehicle, as well as (ii) the time, and (iii) the location of sensing of the relevant training data by the vehicle.
   - The training environment knowledge is clustered into 10 clusters using a K-Means algorithm, based on the location of training data sensing.
   - In each cluster, the vehicle having notified its training environment the most recently is selected for model training at this iteration. In turn, 10 vehicles are selected for training.

   The clustering of vehicles by their last location of sensing of relevant model training data allows to maximize the diversity of training data sensing contexts for the training, to perform the **input variety orchestration** contribution. Moreover, selecting the most recent vehicle whose training environment has been notified to be compatible in each cluster maximized the probability that the selected vehicle has not yet discarded the training data, to perform the **training context availability** contribution.

3. Optionally, an optimization is performed to reduce the number of selected training nodes. In cases where two selected vehicles feature a last known training environments which are too close to each other, one of the selected vehicles is dropped.

4.3.5 Simulation Setup, Results & Discussion

We perform a simulation investigating the potential performance improvements of the VKN-orchestrated approach in terms of training speed and accuracy on a benchmark model, compared with a naive approach which does not take the training environment of vehicles into account.

Namely, we simulate both (i) a set of vehicles in a square area of a fixed size, and (ii) a centralized coordinator, responsible for the orchestration of a FL training process. Periodically, the coordinator selects a set of vehicles to perform an iteration of training. The considered model, which is being trained cooperatively by vehicles and the coordinator is a popular benchmark model, i.e., a classifier for the FEMNIST dataset. We describe the reasons for the choice of the FEMNIST dataset in the next subsection. The vehicles move according to standard mobility models and sense information about their environment as they move. This information is extracted from the training data of the FEMNIST dataset.

Choice of the FEMNIST Dataset

We simulate the FL training by vehicles of a classifier based on the digits-only FEMNIST dataset [168]. It is a FL version of the original MNIST dataset. It associates handwritten digits with the character that is written. In the FL version, the training data is grouped by writer.

In this section, we aim at investigating on the performance gain linked to using VKN to orchestrate and select appropriate training nodes in tasks of *cooperative model training*. The VKN orchestration mechanism of FL training which we implement is independent from the trained model or training algorithm, as it involves selecting relevant nodes for training at each iteration based on their current training environment. As such, the evaluation of the VKN training vehicle selection mechanism requires the flexibility to manipulate the training environment of each vehicle, rather than the choice of a specific model or training algorithm.

In turn, the FEMNIST dataset is a *controllable* and *flexible* dataset where the training data is grouped by author. It allows to replicate a biased distribution of training data where data produced by a single writer is over-represented compared to training data sensed in other contexts, i.e., here, written by other authors. Therefore, as an initial application and proof of concept of the performance gain of VKN *cooperative model training* orchestration, we use the FEMNIST dataset as the model being trained by the simulated vehicles and the centralized coordinator.

Similarly, we use standard random mobility models as in Section 4.2, as we aim to show the potential and impact of VKN through statistically controllable models and

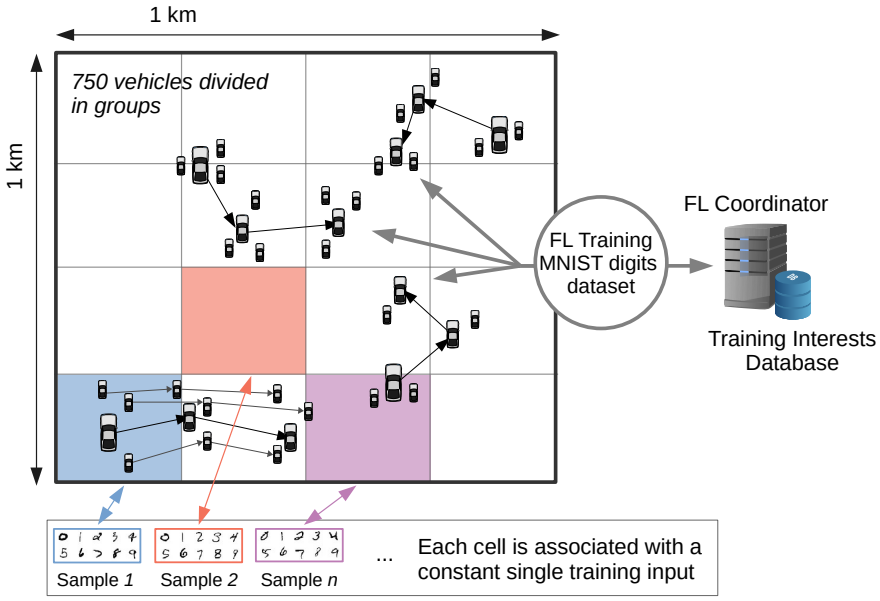


Figure 4.16: Evaluation Setup for Case Study 1 [19]

mobility. Yet, as a second step, a simulation using realistic mobility and knowledge models is required, which we address in Chapters 5 and 6.

General Simulation Setup

We implement two variants of this simulation, matching the two case studies of vehicular mobility and FL training described in Section 4.3.1:

- *Case Study 1* considers the training of a model with commonly observable input. For example, the road surface condition estimation model introduced in [58] matches this condition. In that case, all vehicles in the simulation periodically sense training input for the considered model.
- *Case Study 2* considers the training of a model which requires rarely observed events as training input. For example, the vehicle crash prediction model studied in [59] matches this condition. In that case, only a few vehicles having crossed areas where specific events have occurred will possess the right training environment for the training of the model.

Both case studies are simulated using the same general simulation setup which we describe in this section, with different parameters. Figure 4.16 and 4.17 illustrate, respectively, the specific simulation parameters associated with *case study 1*, and *case study 2*. What is more, the simulation setup for both case studies is described in Table 4.3, and summarized below:

In both case studies, we simulate the FL training of a classifier for the FEMNIST dataset in a vehicular environment. A set of $V = 750$ vehicles are simulated in an area of $1km^2$ surface, over a total simulated time of $T = 1h$. We assume full 1-hop connectivity between each simulated vehicle and a simulated centralized coordinator,

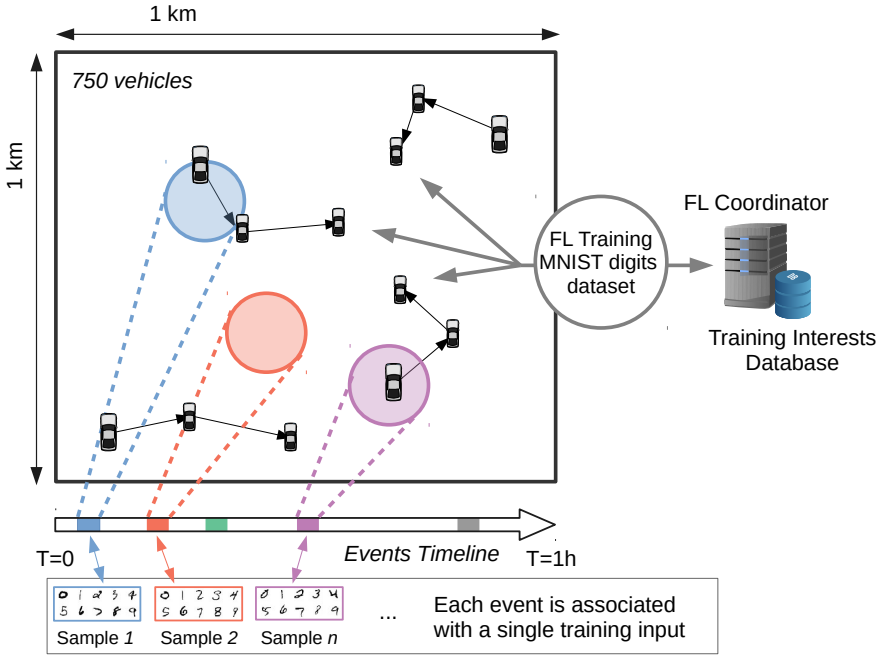


Figure 4.17: Evaluation Setup for Case Study 2 [19]

Table 4.3: Simulation Parameters for the Evaluation of VKN-based FL Orchestration

Parameter	**Value (Case Study 1)**	**Value (Case Study 2)**
Surface of the simulated area	$1km^2$	
Simulated time duration of a simulation	$T = 1h$	
Number of vehicles per simulation	$V = 750$	
Timeout for a training iteration	$15s$	
Number of training vehicles per iteration	10 or less (optimized, see Algorithm 2)	10
Mobility model	RPG	RWP ($P_{pause} = 0.5$, $pause\_max = 5min$, $speed \in [5; 20]m/s$)
Training Input Sensing	Grid-based (see Figure 4.16)	Punctual Events-based (see Figure 4.17)
Training Input Discarding by Vehicles	None (New input sensed every $10s$)	Time relevance (5min timeout), Spatial relevance ($>$ 500m distance)

which is responsible for orchestrating the model training. In practice, the FL training is performed using the TensorFlow Federated library in Python. Specifically, the *Federated Averaging* algorithm is used, which is an implementation of FL.

Mobility & Training Data Sensing

- As illustrated by Figure 4.16, in *Case Study 1*, the mobility model of vehicles is the Reference Point Group mobility model (RPG), to replicate the higher concentration of vehicles in specific points of the simulation, i.e., analogous to main roads. The simulation area is divided into a grid of 10×10 cells of $100m \times 100m$ each. Each cell is associated with a single-author training input extracted from the FEMNIST dataset, to replicate the commonly observable data. Every 10 seconds of simulated time, each vehicle senses the training input which corresponds to the cell it is currently located in, and notifies the coordinator of its updated training environment.

- As illustrated by Figure 4.17, in *Case Study 2*, the mobility model of vehicles is steady-state RWP, with a pause probability of 0.5, a maximal pause duration of 5 minutes, and a speed range of $[5; 20]m/s$. To replicate the sparsity of the training data collection, a timeline of random events is generated. The start time of events are scheduled following a Poisson distribution at a rate of 1.5 events per simulated minute. Each event covers a circular area. Its center position is uniformly sampled in the simulation area, and it has a radius of $r \sim \mathcal{N}(50, 1)$ meters. The duration of each event is sampled from a $\mathcal{N}(60, 10)$ seconds.

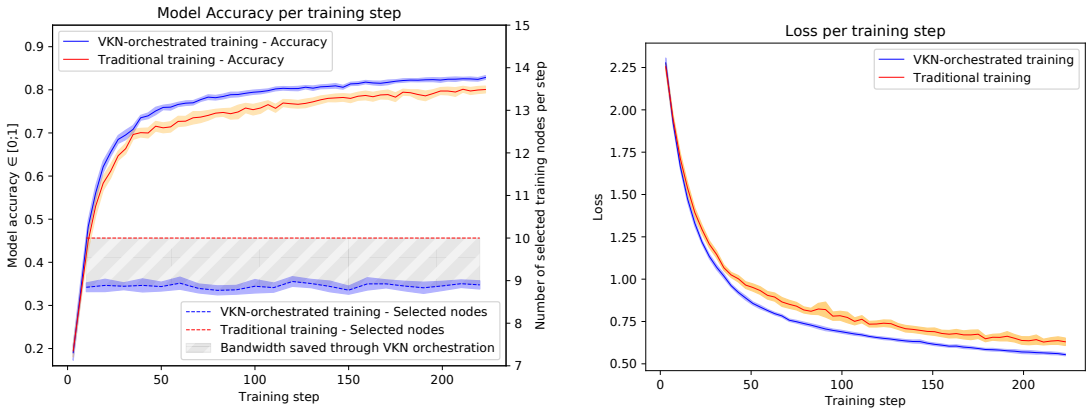
  Each event is associated with a single-author training input extracted from the FEMNIST dataset, and each vehicle whose center point crosses the circular area of a valid event, i.e., started and not expired, will sense the associated training data and notify the coordinator of its updated training environment.

What is more, the simulated vehicles are equipped with data maintenance mechanisms, similar to the LDM maintenance mechanism described in Section 2.5. While this does not impact *Case Study 1*, where fresh training data is sensed every 10 seconds, training data is discarded from each vehicle in *Case Study 2* according to the following rules:

Time relevance If the last sensed training data sample was obtained more than 5min of simulated time ago, it is discarded.

Space relevance If the last sensed training data sample was obtained more than 500m away, it is discarded.

Similarly, the centralized coordinator is equipped with a data maintenance mechanism, which discards training environment knowledge from its KB after a timeout of $\Delta_t = 5min$.



(a) Model Accuracy per Training Iteration (b) Training Loss per Iteration

Figure 4.18: Evaluation of VKN-orchestrated FL Performances for Case Study 1 [19]

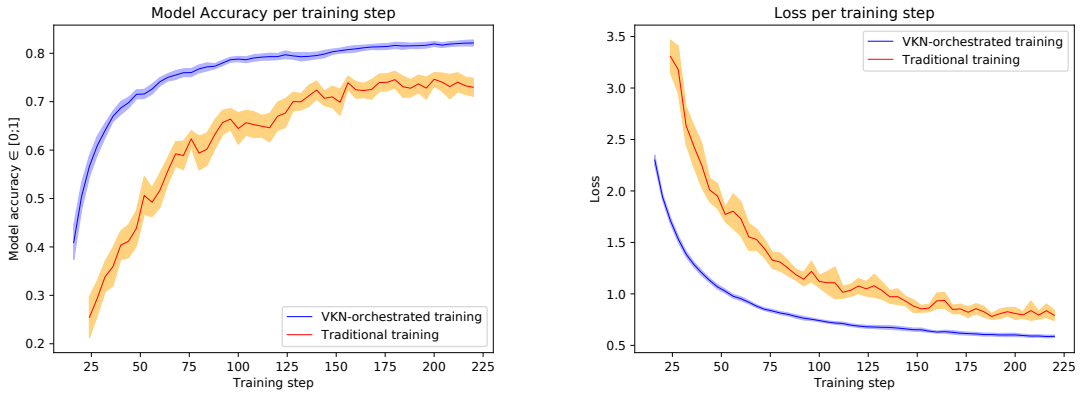
VKN-orchestrated Model Training The training of the FL model is orchestrated by the centralized coordinator associated with each simulation, following the process which we defined in Algorithm 2. In turn, it is compared with a naive approach which selects 10 random vehicles at each training iteration. It is equivalent to Algorithm 2 when $KB.size() == 0$.

4.3.6 Results

In this section, we describe the specific parameters of the simulations performed to represent each case study, and discuss evaluation results.

Case Study 1: Commonly observable input

Figure 4.18 shows the obtained result in terms of model accuracy and training loss per training iteration. The results are bounded by 95% confidence intervals obtained after averaging the results from $S = 25$ simulations. In this case study, orchestration resulted in: (i) a minor but consistent improvement in terms of model accuracy of about 4% while (ii) reducing the amount of nodes selected for training to an average of 9 instead of 10 training vehicles per iteration. Thus, VKN orchestration combines improved training accuracy with reduced bandwidth usage in terms of model update exchange. As a trade-off, vehicles inform the coordinator of their training environment every $\Delta_t = 10s$. Thus, the actual variation of bandwidth usage depends on the trained model and the difference between the size of model updates and that of training environment messages. In complex ANN models encoding numerous weights, the bandwidth saved through the reduction of model update exchanges is likely to be significant over the overhead of training environment messages exchange.



(a) Model Accuracy per Training Iteration (b) Training Loss per Iteration

Figure 4.19: Evaluation of VKN-orchestrated FL Performances for Case Study 2 [19]

Case Study 2: Sparsely Observable Input

Figure 4.19 illustrates the obtained results in terms of model accuracy and training loss per training step. The results are bounded by 95% confidence intervals obtained after averaging the results from $S = 25$ simulations. The orchestration of the FL training through VKN results in significant improvements in terms of convergence time as well as model accuracy. After convergence, the model training through VKN orchestration shows a significance increase in label prediction accuracy of about 15%.

4.4 Discussion

In this chapter, we have contributed the definition of a framework named VKN for (i) knowledge description, (ii) storage, and (iii) dissemination, i.e., the three key aspects identified in Chapter 3. In turn, we defined and performed two initial complementary studies to show the potential and impact of the VKN framework, using simplified but statistically controllable simulations:

- In Section 4.2, we illustrated the impact of the VKN framework on the networking metric of overhead, when knowledge is cooperatively created as a service, instead of systematically distributing full copies of the bytecode of the model.
- In Section 4.3, we illustrated the impact of the networking of rich semantic knowledge and context description content on the accuracy and training speed of knowledge, in two case studies of FL cooperative knowledge creation by vehicles. Both case studies represent the challenges of the vehicular environment related to mobility.

While these initial work items demonstrated initial performance improvements and show the potential impact of VKN both on networking and on the accuracy of knowledge, both studies are preliminary and feature limitations which prevent them from being fully realistic.

To begin with, in Section 4.2, we use a RWP standard random mobility model to simulate the mobility of vehicles. As it is statistically controllable, it is appropriate to show a tendency and potential of VKN. Yet, it does not realistically represent the mobility of vehicles. What is more, we perform a model-agnostic study, with a simplified theoretical passenger comfort estimation model as a case study. In turn, the definition of realistic knowledge for a real use case is necessary to confirm the observed potential of VKN.

Similarly, in Section 4.2, we use the RPG and RWP standard mobility models, statistically controllable but not realistic regarding real vehicular mobility. What is more, while the contribution of Section 4.2 lies in the algorithm of training vehicle selection for FL training, i.e., model agnostic, we use the FEMNIST model as a benchmark model to show the impact of the VKN-supported training vehicle selection. In turn, the FEMNIST model of digit recognition does not represent a realistic vehicular situation.

As such, while both studies show the relevance of the framework and potential performance improvement, a study is required which uses realistic vehicular mobility data, as well as knowledge, to complete these initial studies. What is more, in this chapter, the impact of using knowledge which has the right interface but in the wrong context was not evaluated.

In Chapters 5 and 6, we contribute the definition of new driving risk knowledge in roundabout as a realistic vehicular knowledge model. In turn, we contribute an extraction of the semantic features which can be used to describe the relevant context of application of this risk knowledge. Then, we evaluate, through a network simulation, the impact of using knowledge in the right context, using its semantic description.

Chapter 5

Definition of a Roundabout Risk Knowledge Application for VKN

In Chapter 4, we have introduced VKN as a concept framework to integrate knowledge description to the storage and distribution of knowledge, and showed initial performance improvements in generic applications of knowledge as a service and cooperative knowledge training through federated learning. These initial studies, described, respectively, in Sections 4.2 and 4.3, illustrate the potential and the impact of the VKN framework for knowledge distribution on vehicular networks. Yet, they were performed using simplified random mobility models, as well as non-vehicular-related knowledge, chosen for their flexibility and statistic controllability, which allowed to show the potential of VKN. In turn, studies featuring real vehicular mobility, as well as realistic knowledge as part of a vehicular use case, are required to truly evaluate the performance of VKN in real applications.

In turn, in Chapters 5 and 6, we define a realistic knowledge model applied to driving risk in vehicular networks and evaluate its distribution supported by the VKN framework, using real vehicular mobility traces. Namely, in this chapter, we define and contribute knowledge for driving risk estimation for Connected and Autonomous Vehicles (CAVs), especially in roundabouts. The architecture of this chapter is as follows:

1. To begin with, we define a use case of driving risk-based navigation for connected and autonomous vehicles.

2. In turn, we summarize the existing known factors of risk for the self-driving features of a CAV.

3. In a preliminary study, we evaluate the impact of the extreme case of strictly rerouting CAVs away from all risk factors.

4. Then, we contribute the definition of a dynamic driving risk knowledge, in the specific case of roundabouts, to allow an adaptive, rather than strict, risk-based routing of CAVs.

5. Finally, we open perspectives on the impact of the driving risk knowledge we contributed for future CAV risk-based routing applications.

In Section 5.1, a preliminary case study is described which illustrates the need for defining driving risk knowledge for autonomous driving applications. It covers the items 1, 2, and 3 of the aforementioned chapter architecture. Secondly, we study real vehicle tracks extracted from roundabouts to define a model of risk knowledge on roundabouts in Section 5.2, which corresponds to item 4. Finally, we open perspectives for item 5 in a discussion part. In Chapter 6, we use the roundabout exit probability estimation risk model defined in Section 5.2 to evaluate the performance of VKN on a realistic vehicular driving risk application.

5.1 Case Study: Driving Risk-Aware Navigation

In this chapter, we consider the use case of driving risk-based navigation for connected and autonomous vehicles. Namely, we consider the ability for a CAV to choose routes to go from a departure point A to a destination B while minimizing its exposure to driving risk situations, and specifically situations which challenge its ability of self-driving.

To begin with, we investigate the environmental and traffic-related impact of the introduction of CAVs on the road, to motivate the need for the definition of risk knowledge to support CAV navigation. The introduction of CAVs on roads is expected to improve the fluidity of traffic, as they have a potential to exchange information to coordinate their movements and take cooperative navigation choices which improve the overall flow of traffic. According to [169], the introduction of connected vehicles at a penetration rate as little as 6% can efficiently mitigate the formation of 'shockwaves' of vehicles slowing down due to congested traffic conditions. What is more, [170] described an architecture to delegate the global routing decision of CAVs to a centralizer controller unit. In turn, by selecting the routes of each vehicle with the aim of improving traffic flow, the mean travel time and traffic throughput in a simulation of downtown Toronto were improved.

As such, the route choices of CAVs can be constrained to reach improvements in traffic flow metrics. Yet, this supposes strictly following the routes which were assigned to each vehicle. In some cases, this might work against the interests of CAVs, e.g., because of an override of the route of a CAV commanded by one of its passengers, or because some roads might challenge the functioning of the automated driving capabilities of a CAV. Namely, route synchronization mechanisms which aim at improving traffic flow might require the CAVs to cross various roads, i.e., primary, secondary or smaller roads, and intersections, i.e., traffic lights-based but also yield-based, such as roundabouts. In turn, the design of many historical roads did not take ease of navigation for CAV into account. On the contrary, CAVs must adapt to increasingly complex road infrastructures which challenge the self-driving algorithms on-board CAVs.

If the self-driving algorithms on board a CAV reach a capacity limit due to the crossing of such areas, the vehicle is at risk for unsafe behaviors such as a slow down

or complete stop of the CAV. Alternatively, CAVs might decide to return the control to a human supervising driver, in a *take-over* procedure, which might constitute a strong risk if, as considered in [171], the driver suffers from a lack of attention at the critical moment of take-over.

As such, instead of strictly following the optimal route choices which were assigned to them to globally increase traffic flow, CAVs may prefer alternative routes which are less likely to pose a safety threat to their self-driving features and passengers. Namely, [172] investigated the routing of CAVs away from *risky* areas, following a risk metric based on the occlusion of their field of view at certain intersections. This causes the CAVs to be *de facto* routed away from certain areas.

In this preliminary study, we consider the impact on traffic and on the environment of systematically routing CAVs away from certain *risky* areas, to minimize the safety risk for passengers of CAVs in their current state of automation. Finally, we show that the definition of risk knowledge is required, to let CAVs dynamically assess the level of risk in a specific area rather than avoiding it altogether, to ensure safety while offering some degree of flexibility in the route choices of CAVs.

5.1.1 Summary of Contexts of Risk for Self-Driving Capabilities

A widely accepted scale detailing the levels of automation of CAVs has been defined as part of the SAE J3016 standard [173]. It starts at level 0, i.e, no automation, to level 5 for full automation, i.e., the vehicle can drive in every road and situation without requiring the intervention of a human driver. CAVs which are currently being tested typically operate at a level 3 or 4 of automation, in which the system can control the vehicle automatically in a set of known situations. A supervising human driver must be present in the vehicle.

As such, certain roads or driving contexts might lead to a capacity limit of the automation algorithms on-board a CAV, triggering a slow down, complete stop, or a take-over by the supervising human driver. In this section, we provide an overview of situations which are known to be challenging to negotiate autonomously for CAVs.

We perform a summary of the literature in order to determine which areas imply such risk or perturbation for current automated vehicles. We divide the problem into two aspects, namely (i) *topological risk*, where the shape of the driveable way itself might challenge self-driving units, and (ii) *human risk*, where the need of a complex interaction with human-driven vehicles causes disruption. Table 5.1 summarizes the risk factors for the self-driving capabilities of CAVs which we describe in this section.

Topological Risk

A description of (i) the driving conditions which are known to be challenging to address for CAVs and (ii) the reasons which are typically considered to make a CAV trigger a human take-over are summarized, respectively, in [176] and [177].

Table 5.1: Summary of Risk Factor for the Self-Driving Features of CAVs

Risk Factor	**Possible Causes**	**Reference**
Topological Risk		
Low visibility	Occlusion by buildings or other vehicles, weather conditions	[172, 174], [175]
Complex road layout	Unclear ground markings, work zones	[176]
Complex maneuvers	Lane change in congested traffic	[177]
Capacity limit	Too many vehicles and VRUs, congested traffic	[177]
Human Risk		
Understand the intentions of legacy vehicles	Unsignalized yield intersections, e.g., give-way, roundabouts	[178, 179]
Understand the intentions of VRUs	Erratic behavior, unexpected pedestrians	[180]
Communicating CAV intentions to human road users		[181]
Unpredictable behavior of human road users towards CAVs		[181]

As described in [176], CAVs highly depend on the information sensed by their on-board sensors to implement self-driving. CAVs come equipped with a variety of sensors, including cameras, radars, or LiDARs. As such, the self-driving capabilities of a CAV can be challenged both by (i) situations which alter the quality of the information produced by sensors, and (ii) situations which are too complex for accurate decisions to be taken based on the current scene.

On the one hand, for the first aspect, the challenge of poor visibility is listed as a key challenge. Namely, reduced visibility is likely to reduce the coverage and quality of the information produced by sensors, and in turn alter the perception of the environment of a CAV. This applies to occluded intersections, e.g., because of buildings, as described in [172] and [174]. What is more, poor weather conditions, as specifically described in [175], may also deteriorate the quality of sensor measurements.

On the other hand, the road layout can be too complex to be accurately interpreted by the on-board unit of a CAV, for example, while crossing work zones which are not documented in the map data, or generally in case of unclear ground markings. In particular, work zones in which both the temporary and the regular ground markings can be seen are challenging for CAVs.

In [177], the driving contexts which are likely to trigger a human take-over which are described are also related to a capacity limit of the scene perception algorithms of CAVs. A capacity limit is reached when one or several of the following events occur:

- A complex maneuver has to be performed, such as a lane change.
- Weather conditions affect visibility.
- Occlusion affects visibility.
- The number of vehicles and Vulnerable Road Users (VRUs) on the road increases past a threshold.

Namely, [177] classifies situations of congested traffic in which the number of vehicles to consider is high as a factor of risk for CAV take-over. Unexpected behaviors of other road users is also listed as a challenge, especially considering the

interaction with two-wheelers and pedestrians, which can trigger a take-over if their density is too high.

Human Risk

The interaction between CAVs and human-driven vehicles is a key challenge for the implementation of self-driving systems. Specifically, algorithms are required to understand the intentions of human road users, as described in [178, 179]. This is especially true in unsignalized intersections which require negotiation with human beings, such as roundabouts. Even without human-driven vehicles on the road, cyclists and pedestrians must still be considered, and their intentions, e.g., to cross or not cross, must be understood. On the other hand, CAVs, which eventually aim to be driverless, must develop means to communicate their intentions to legacy human-driven cars, or non-connected VRUs.

As such, events which are critical for CAV safety were defined in [180]. The unpredictable and potentially erratic behavior of other human-controlled road actors as well as the unexpected presence of pedestrians in roads or locations where they are not supposed to be are listed as situations which make the human-machine interaction challenging, and might trigger a take-over of CAVs. Specifically, unsignalized yield intersections, i.e., stops, give way and roundabouts require negotiation with human drivers, especially in congested traffic conditions, and may constitute a risk of take-over.

What is more, the challenges related to the interaction between CAVs and pedestrians have been surveyed in [181]. A key issue is the understanding by CAVs of the intentions of pedestrians. It is also noted that pedestrians may behave differently when recognizing that the vehicle they are interacting with is a CAV, potentially leading to a lack of accuracy of pedestrian behavior prediction models trained without the presence of self-driving vehicles.

5.1.2 Impact of Strict Risk-Based Routing

Based on the criteria described in Section 5.1.1 and summarized in Table 5.1, we investigate the impact on traffic and on the environment of routing CAVs away from areas of potential risk. Namely, we identify a list of *risky* areas for self-driving vehicles in Monaco. In turn, we route CAVs away from these *risky* areas in a microscopic traffic simulation of the city-state. We deliberately route vehicles strictly away from the areas of risk, to eventually show the need for the building of a finer risk knowledge, which would allow to route vehicles based on a dynamic assessment of the risk in various areas of a city.

Risk Area for CAV in Monaco

We manually identify a list of intersections in the city of Monaco which are likely to be challenging to drive for CAVs, as they match a combination of criteria extracted

Table 5.2: Summary of the Considered Risky Zones

	Type	Occlusion	Pedestrians	Other
Rond point du portier	Roundabout		✗	Work zone
Rond point Place D'armes	Roundabout	✗	✗	
Boulevard des moulins	Roundabout	✗	✗	Unclear marks
Boulevard d'Italie	Roundabout			Unclear marks
Avenue de la Madone	Yield	✗	✗	
Avenue de France	Yield		✗	Lane merging
Pont Sainte-Dévote	Yield			Lane merging

from Section 5.1.1 and Table 5.1. In particular, we consider (i) the type of intersection, e.g., roundabout, (ii) the amount of occlusion, (iii) whether an unsignalized pedestrian crossing and/or lane merging is present, (iv) the clarity of ground marks, and (v) the typical traffic conditions in the morning rush hour, as we run microscopic traffic simulations of Monaco for that period.

In total, we list seven intersections of high risk for CAV self-driving systems. Their location is summarized in Figure 5.1, and their characteristics, in Table 5.2. A precise description of the rationale behind the selection of each of these intersections in given in the associated research report in [20].

Simulation Setup

We run a series of simulations replicating the traffic of a typical morning rush hour in Monaco. In this simulation, we flag a fraction of vehicles as 'CAVs', and evaluate the impact of a strict risk-based routing of CAVs away from the risky intersections identified in Table 5.2. The rest of the vehicles is flagged as 'human-driven vehicles' and their routes are not recomputed to avoid the aforementioned areas. In order to demonstrate the need for the definition of a finer risk knowledge which can be dynamically computed for each intersection, we deliberately implement an extreme case where CAVs are systematically routed away from the risky areas, in order to avoid any potential risk, as no risk knowledge is in use.

The simulations are implemented using the Simulation of Urban MObility (SUMO) tool, which is a general purpose microscopic traffic simulator [182]. This means that each vehicle of the simulation is independently simulated. Through its Traffic Control Interface (TraCI) APIs, the mobility of simulated vehicles can be controlled, which lets us implement the scenario of routing CAVs away from risky areas. To simulate the traffic of a morning rush hour in Monaco, we use the Monaco SUMO Traffic (MoST) scenario, as defined in [183]. It is a realistic simulation of the morning mobility in Monaco from 5:00 to 12:00. What is more, MoST is a model of Monaco in three dimensions, including elevation data. Due to its location, Monaco features high elevation differences among areas, and high slopes for some roads. In turn, we use the *PHEMLight* [184] emissions model, a simplified version of the *Passenger Car and Heavy Duty Emission Model* developed by TU Graz, which takes the slope of the road into account when computing the emissions produced by each vehicle.

MoST contains multiple modes of transportation, including passenger cars, buses, trains, pedestrians, and two-wheelers. Overall, one run of the MoST simulation in-

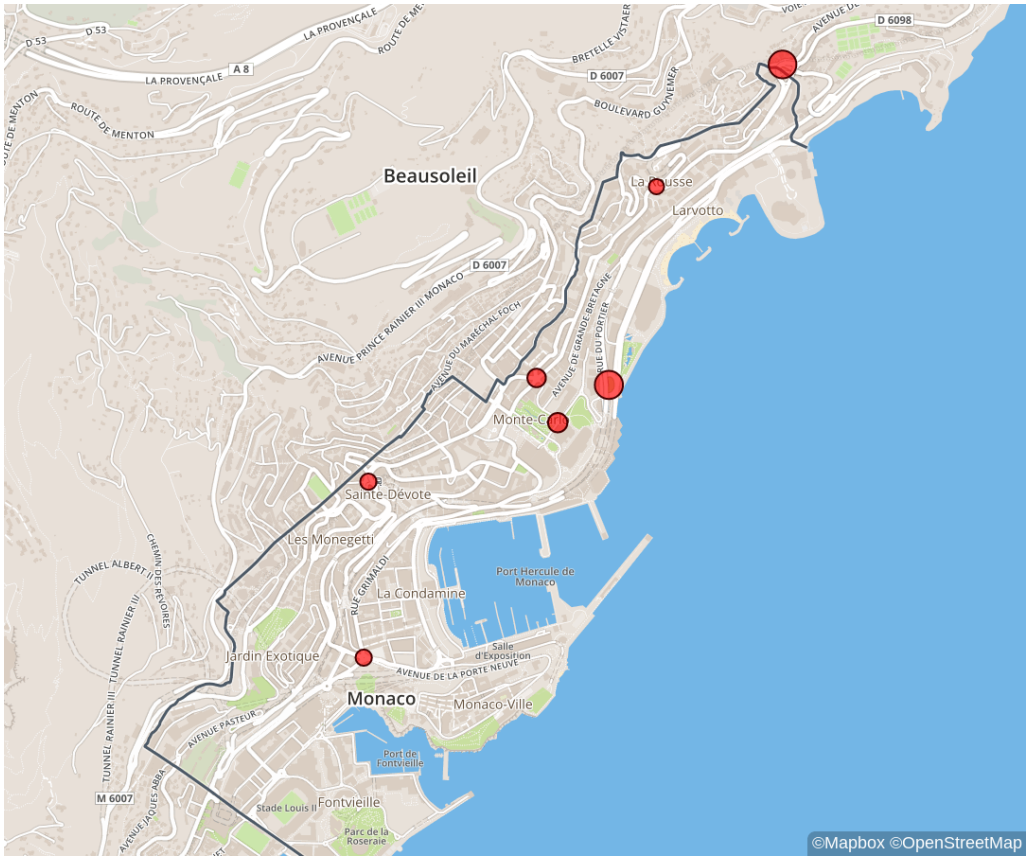


Figure 5.1: Risky Areas Considered for CAVs in Monaco

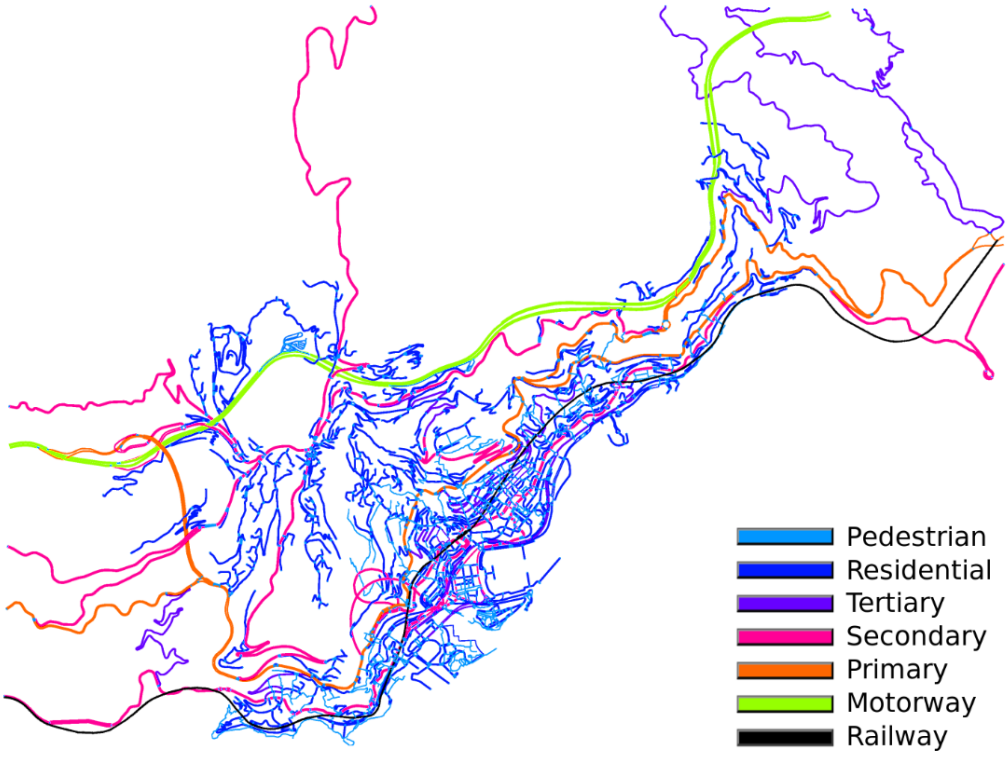


Figure 5.2: Overview of the MoST Scenario [183]

volves about 22700 passenger vehicles, which all feature a starting point and a destination. While each vehicle has an initial position and a destination before leaving the simulation, the routing of each vehicles is not fixed. While it is precomputed to the shortest path using the Dijkstra algorithm, the route of each simulated vehicle is reevaluated every P=300s of simulated time, to be updated to the new shortest path to the destination, taking the current traffic conditions into account. Specifically, the cost of each path is computed as its overall distance over the mean speed of vehicles which currently drive through the path.

In our simulations, we assign each passenger vehicle to a virtual category. Each vehicle is either flagged as 'human-driven' or as a 'CAV':

- The behavior of 'human-driven' passenger vehicles is left untouched. They follow the standard MoST behavior of reevaluating their route every 300s to take the fastest path to their destination, given current traffic conditions.
- Passenger vehicles which are flagged as 'CAV' behave like 'human-driven' vehicles, with the exception that, in order to minimize their likelihood of breaking self-driving features, the routes that their reevaluate every 300s will never include a crossing of any of the intersections which we flagged as risky for CAVs in Table 5.2.

We run several simulations in which we vary the proportion of passenger vehicles which are flagged as CAVs and subsequently routed away from the risky areas listed in Table 5.2. Namely, for each value of CAV proportion $r_{CAV} \in [0, 10, 20, 35, 50, 60]\%$, $n = 10$ simulations are run with different random assignment of 'CAV' and 'human-driven' flags, in order to compute confidence intervals.

We collect the following metrics from each completed simulation:

- General traffic flow statistics at each step of the simulation such as the mean speed of vehicles.
- Statistics about the overall length of queues throughout the simulation.
- Vehicle-wise trip data, including their trajectory and fuel consumption.
- Pollutant emission statistics for each vehicle, including CO_2, CO, and NO_x gases, PM_x particles, and hydrocarbon emissions. Passenger vehicles are gasoline-powered and follow the EURO 4 emission standard.

In turn, we analyze the results obtained from each of the 10 simulations, for each value of CAV proportion $r_{CAV} \in [0, 10, 20, 35, 50, 60]\%$.

Evaluation Results

Routes To begin with, we analyze the impact of routing 'CAV' vehicles away from the risky areas defined in Table 5.2 in terms of the different routes which are taken by 'CAV' and 'human-driven' vehicles. Figure 5.3 illustrates the difference in the

routes taken by 'CAV' and 'human-driven' vehicles for simulations which involve an equivalent amount of each type of vehicle, i.e., $r_{CAV} = 50\%$. For each road section, the difference between the number of 'CAV' and the number of 'human-driven' vehicles which have crossed the road over a complete simulation is computed and normalized in the $[-1, 1]$ interval. In turn, blue-colored roads represent roads which were mostly occupied by 'CAV' vehicles, whereas red-colored roads were preferred by 'human-driven' vehicles. As such, we notice a clear impact of the strict risk-based routing of 'CAV' vehicles on their choice of route. Namely, the main roads which lead to downtown Monaco are overwhelmingly crossed by 'human-driven' vehicles, whereas 'CAV' vehicles shift to secondary roads to avoid the risky intersections defined in Table 5.2.

Furthermore, the increase of the proportion r_{CAV} of 'CAV' vehicles on the road leads 'human-driven' vehicles to increasingly favor the main and primary roads, as they are left free by 'CAV' vehicles. Figure 5.4 illustrates the difference in the roads which are the most driven on by 'human-driven' vehicles, respectively, when no 'CAV' vehicles are present in red, i.e., $r_{CAV} = 0\%$, and when a majority of vehicles are 'CAV' in blue, i.e., $r_{CAV} = 60\%$. Namely, a scale of popularity of roads driven by 'human-driven' vehicles is defined in the $[0, 1]$ range, 0 being a road which no 'human-driven' vehicle crossed during a complete simulation, and 1 being the road which the most 'human-driven' vehicles have crossed. In turn, Figure 5.4 shows that 'human-driven' vehicles adapt to the decreasing amount of vehicles in primary roads as r_{CAV} increases and 'CAV' vehicles leave them free.

This significant difference in routes between 'CAV' and 'human-driven' vehicles, as well as the adaptation of 'human-driven' vehicles to the spaces left free by 'CAV' vehicles is likely to have an impact on the traffic flow and emission statistics throughout a simulation. In the next paragraph, we consider the impact of the strict risk-based routing of 'CAV' vehicles away from risky intersections defined in Table 5.2 on trip statistics for passenger vehicles.

Trip duration Figure 5.5 illustrates the evolution of the mean trip duration of passenger vehicles in a simulation, as the proportion of r_{CAV} 'CAV' vehicles increases. The mean trip duration of 'CAV' and 'human-driven' vehicle are shown, respectively, in blue- and orange-colored lines. Then, the green-colored line shows the mean trip duration for all passenger vehicles in the simulation, regardless of their type. Each value is associated with a 97.5% confidence interval, computed from $n = 10$ simulations. We notice that the increase of 'CAV' vehicles, which are strictly routed away from risky intersections, has little impact on the overall mean trip duration of vehicles. However, it negatively impacts the trip time of 'CAV' vehicles, as the secondary roads they take get more congested. On the other hand, the trip time of 'human-driven' vehicles dramatically decreases as the primary roads they cross get less congested.

Similarly, Figure 5.6 illustrates the evolution of the mean speed of vehicles in a simulation, for various values of r_{CAV}. We notice similar effects as for the trip time, i.e., while the overall mean speed remains stable, the speed of 'human-driven' vehicles increases and that of 'CAV' vehicles decreases when r_{CAV} increases. This could be explained similarly by the formation of potential queues on secondary roads which

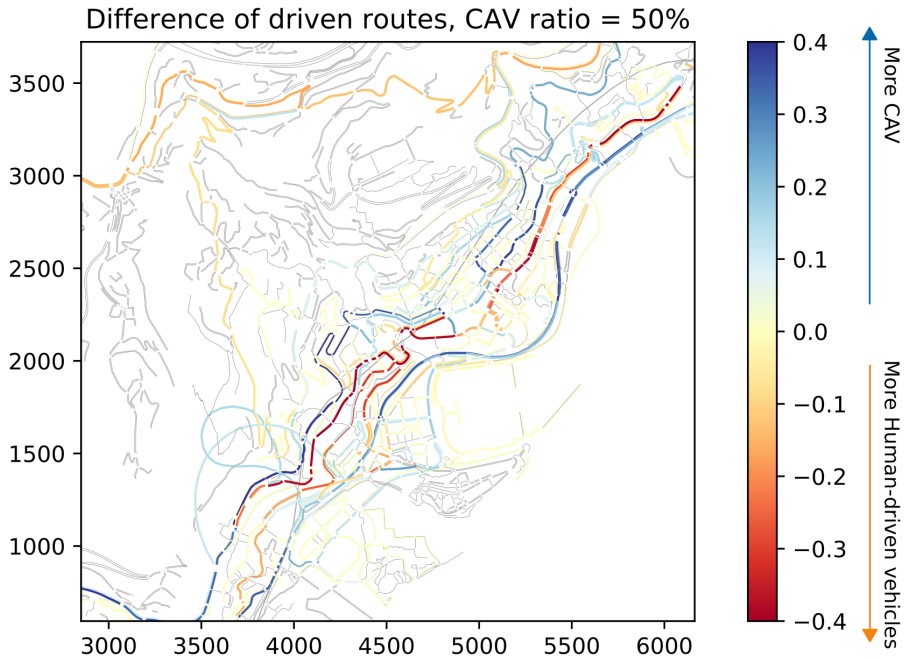


Figure 5.3: Difference of Routes between CAV and Human-driven Vehicles for a CAV Ratio of 50%

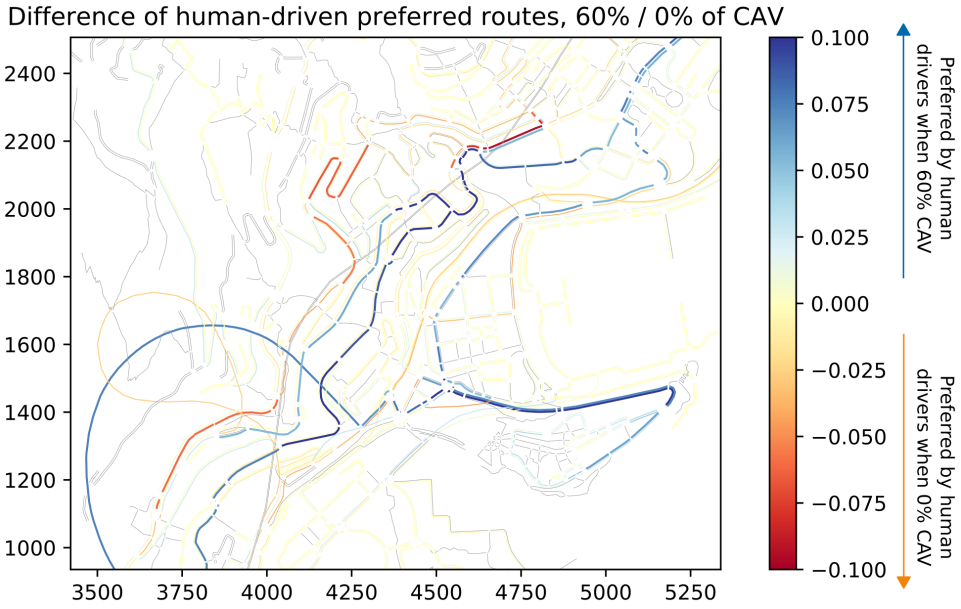


Figure 5.4: Difference of Preferred Routes by Human-driven Vehicles for CAV Ratios of 0 and 60%

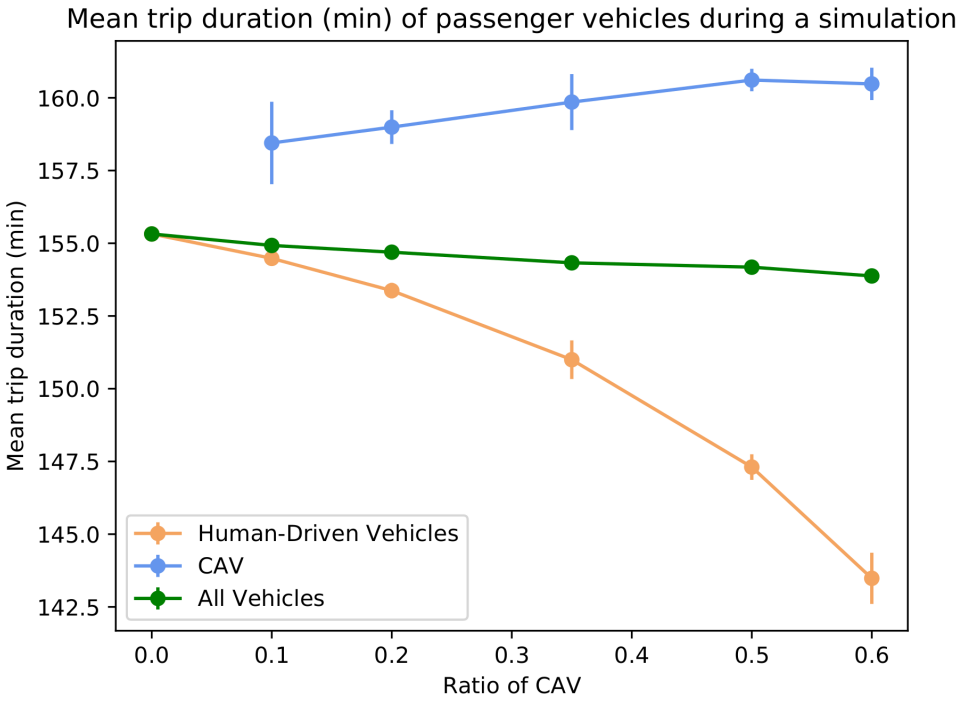


Figure 5.5: Mean Trip Duration of Vehicles during a Simulation

are favored by 'CAV' vehicles. In the next paragraph, we investigate the impact of strict 'CAV' risk-based routing on the formation of queues in the simulations.

Queues and congestion Figure 5.7 illustrates the evolution of the accumulated length of queues on the roads of a simulation for each time step, for various values of proportions of 'CAV' vehicles r_{CAV}. For each line, a 95% confidence interval is computed from the $n = 10$ associated simulations. We notice a significant increase in the length of queues during the peak rush hour time, i.e., 8:30 to 9:00, starting from $r_{CAV} = 20\%$ included. In parallel, Figure 5.8 illustrates the mean *relative* speed of all passenger vehicles throughout a simulation. The mean relative speed is the speed of a vehicle normalized by the maximal allowed speed on the current road. Similarly, we notice a significant reduction of the mean relative speed of vehicles for $r_{CAV} > 20\%$, during the rush hour peak time.

Thus, after a given threshold of 'CAV' vehicles, located between $r_{CAV} = 10\%$ and 20%, the extra amount of vehicles on secondary roads due to the strict risk-based routing of 'CAV' vehicles leads to an increase in queue length and a simultaneous reduction of the mean driving speed. In turn, this potentially increases the emission of pollutants during peak time, as we analyze in the next paragraph.

Pollutant emissions On the one hand, Figure 5.9 illustrates the evolution of the mean emissions of CO_2 by driven distance in g/km, by, respectively, (i) 'CAV' vehicles in a blue-colored line, (ii) 'human-driven' vehicles in an orange-colored line, and (iii) all passenger vehicles, regardless of their type, in a green-colored line. Each

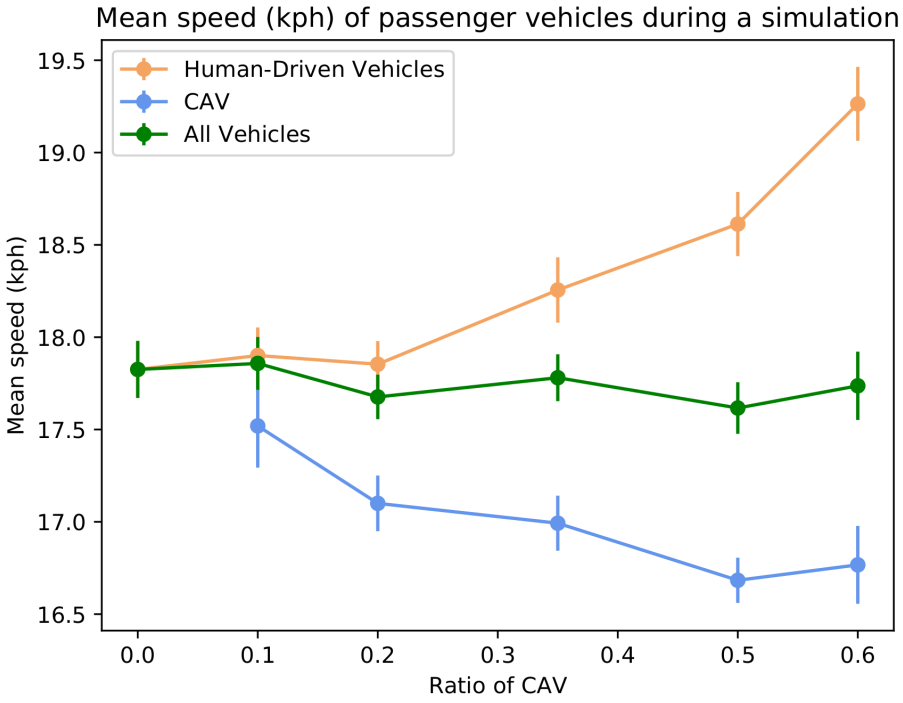


Figure 5.6: Mean Speed of Vehicles during a Simulation

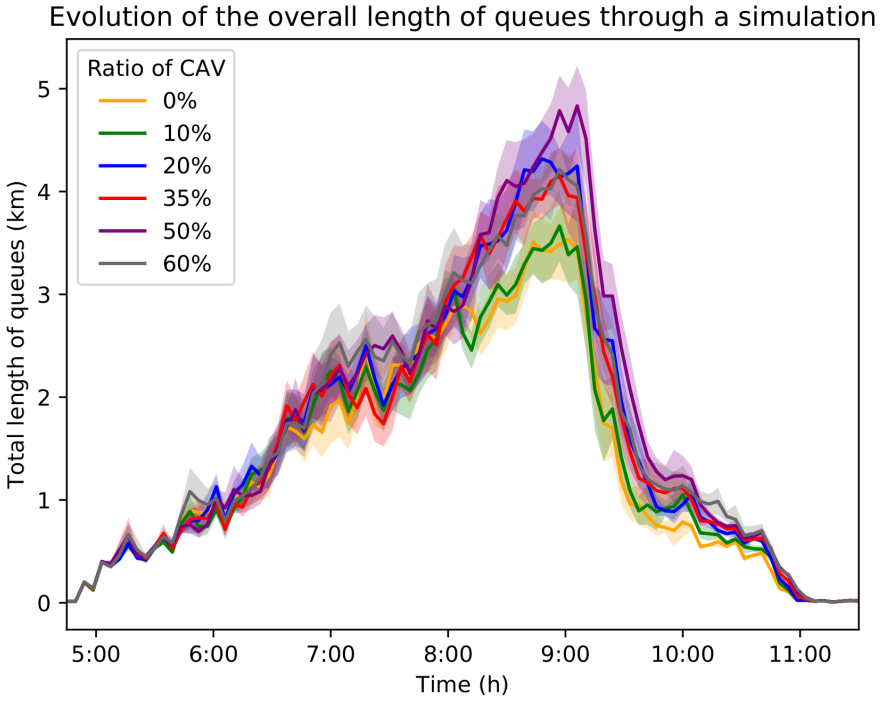


Figure 5.7: Evolution of the Overall Queue Length

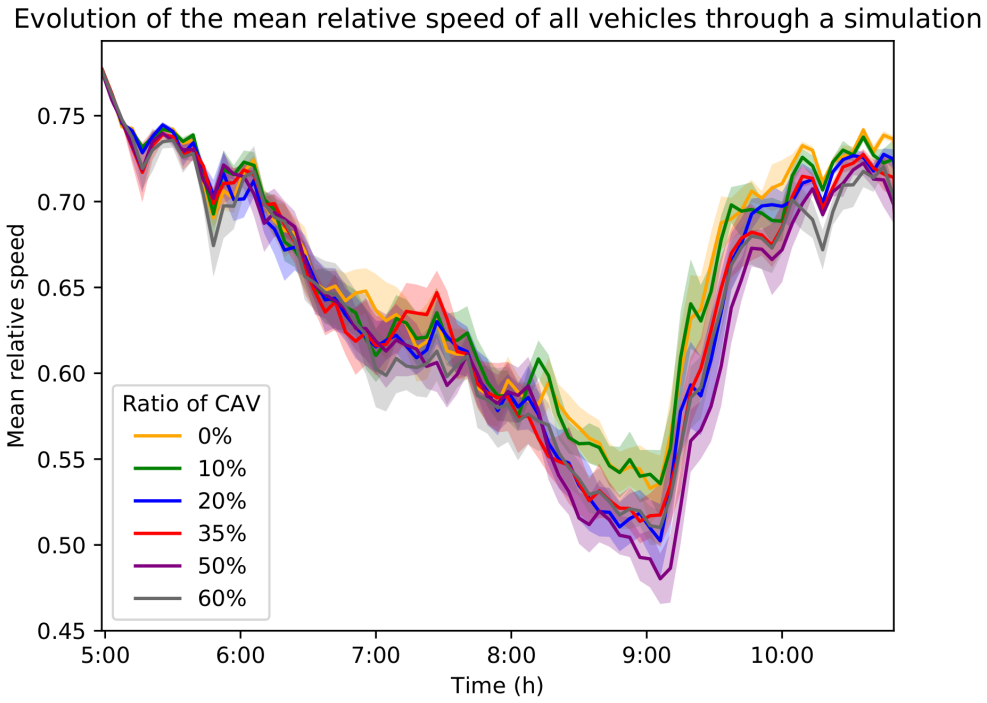


Figure 5.8: Evolution of the Mean Relative Speed of Vehicles

point features a 97.5% confidence interval computed from the associated $n = 10$ simulations. We observe a significant and stable 5% increase in the emissions per kilometer of 'CAV' vehicles compared to 'human-driven' vehicles. In turn, when all vehicles are considered, the emissions of CO_2 grow with the proportion of 'CAV' vehicles r_{CAV}. Similar results are observed for other pollutants. Namely, 'CAV' vehicles emit 7% more NO_x gases, 6% more CO gases and HC hydrocarbons, 5% more PM_x particles, and burn 5% more fuel per kilometer than 'human-driven' vehicles. While electric vehicles could mitigate these emissions, the obtained results demonstrate an increased use of energy by 'CAV' vehicles as they strictly avoid risky areas defined in Table 5.2.

On the other hand, Figure 5.10 illustrates the global CO_2 emissions in metric tons in simulations for the considered values of r_{CAV}, with 97.5% confidence intervals. In the bottom part of the figure, the global emissions are divided between 'human-driven' vehicles in orange and 'CAV' vehicles in blue. The dark blue area represents the excess emissions of 'CAV' vehicles compared to their proportion in the simulation. Similarly to the queue length results illustrated in Figure 5.7, we notice a significant increase in the overall CO_2 emissions after a threshold of $r_{CAV} = 20\%$ of 'CAV' vehicles in the simulation, included. Then, the emissions further increase starting from $r_{CAV} = 50\%$. The emissions of NO_x as well as fuel consumption of vehicles follow a similar pattern, while the emissions of CO gases, PM_x particles and hydrocarbons grow linearly with r_{CAV}.

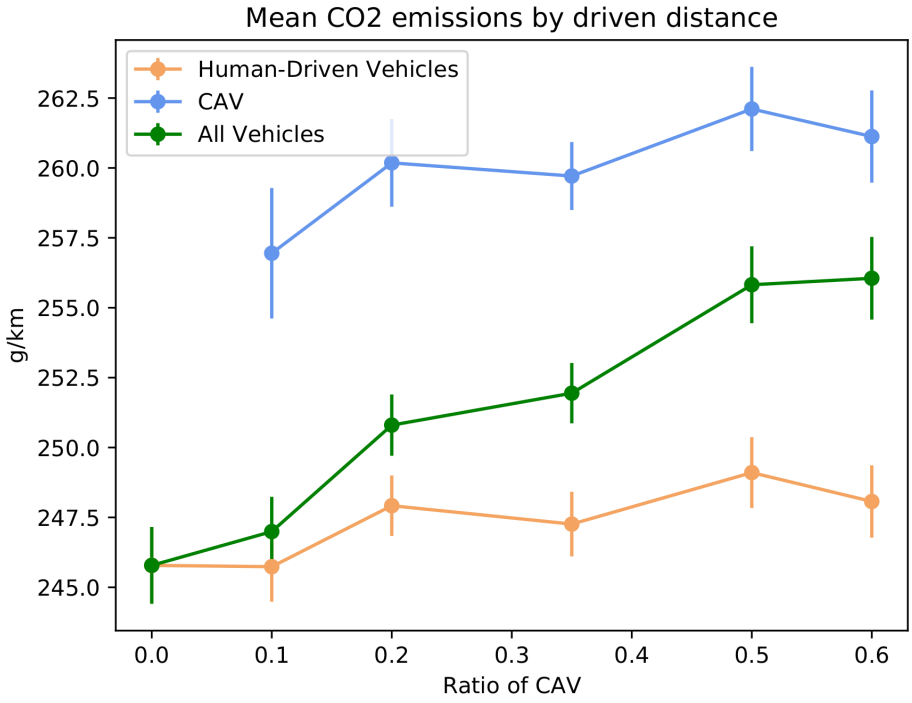


Figure 5.9: Evolution of the Emissions of CO_2 by Driven Distance by Vehicle Type

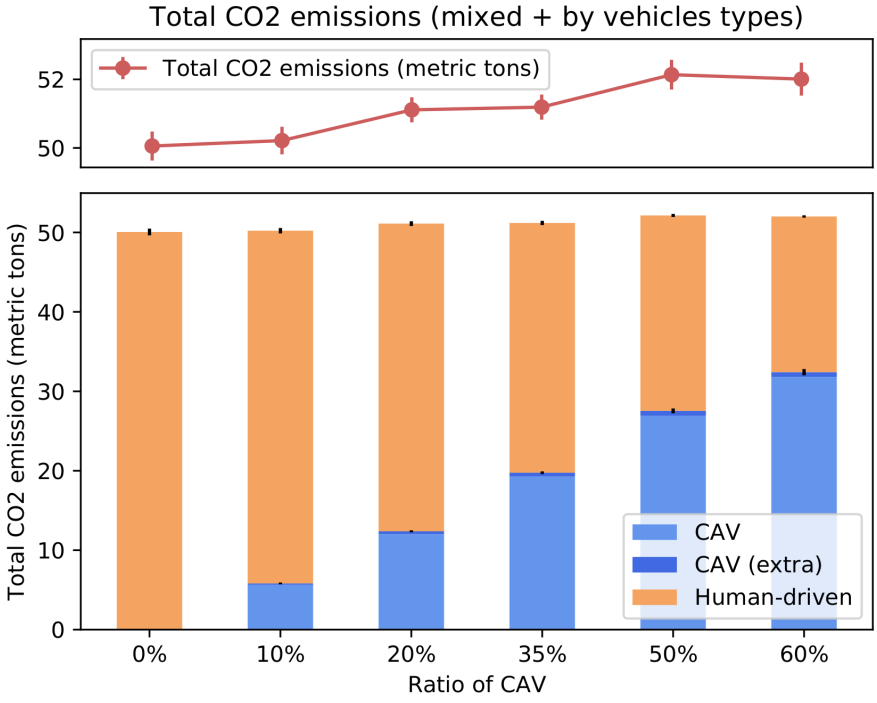


Figure 5.10: Total CO_2 Emissions by Vehicle Type in a Simulation

5.1.3 Discussion

While CAVs have the potential to improve traffic fluidity through coordinated route choices, a key challenge of CAVs in their current level of automation is to mitigate the risk to break their self-driving algorithms and trigger a risky take-over by a human supervisor. In turn, CAV routes may be chosen so as to avoid areas which put the self-driving algorithms at risk, as suggested in [172].

In this work item, we deliberately implement a strict and extreme behavior for CAVs to systematically avoid the intersections of potential risks, to reduce the likelihood for a take-over to be triggered. In Section 5.1.2, we identified a set of intersections which accumulate risk factors for the self-driving algorithms of CAVs in the city of Monaco, as listed in Table 5.2. In turn, we perform a simulation where a fraction of vehicles is flagged as 'CAV' and routed, albeit dynamically, to systematically avoid the aforementioned risky areas.

The results show a degradation of the trip time of 'CAV' vehicles, compensated by an improvement for legacy 'human-driven' vehicles. Yet, overall, the formation of queues and emissions of pollutants in the simulation increases with the proportion of 'CAV' vehicles, as they avoid areas of risk to focus on secondary roads to reach their destination. As such, the potential benefits of CAVs route coordination in terms of traffic flow were voided by a strict choice to mitigate the risk of take-over triggering by 'CAV' vehicles.

Yet, even in the intersections which we list in Table 5.2, the level of risk is not constant, unlike considered in this study. Instead, what if *driving risk* knowledge could be dynamically computed and updated with time for each intersection, to let 'CAV' vehicles cross them if the current level of risk is low? Namely, the definition of risk knowledge could bring a trade-off between safety and the benefits of 'CAV' vehicles for traffic fluidity, studied in [170].

As such, in the next section, we study and define a form of risk knowledge applied to the specific case of roundabout intersections. Provided with this study on roundabout driving risk knowledge building, we investigate the distribution of this knowledge in vehicular networks in Chapter 6, to pave the way for a routing of CAVs based on the VKN dissemination of driving risk knowledge.

5.2 A Roundabout Risk Knowledge Definition

Based on the preliminary study, we choose to focus on roundabouts and aim at defining risk knowledge for roundabouts. In turn, we investigate on roundabout risk knowledge sharing through VKN to pave the way for CAVs to adapt their behavior to the current level of risk in a roundabout.

As introduced in Section 5.1, a roundabout is a yield intersection, which as such may require negotiation with other road users, and understanding the intentions of other vehicles to be crossed safely. Existing works have considered the safety impact of the introduction of CAVs to roundabout traffic, using calibrated micro simulations [185, 186, 187]. While the introduction of CAVs has reduced the amount of

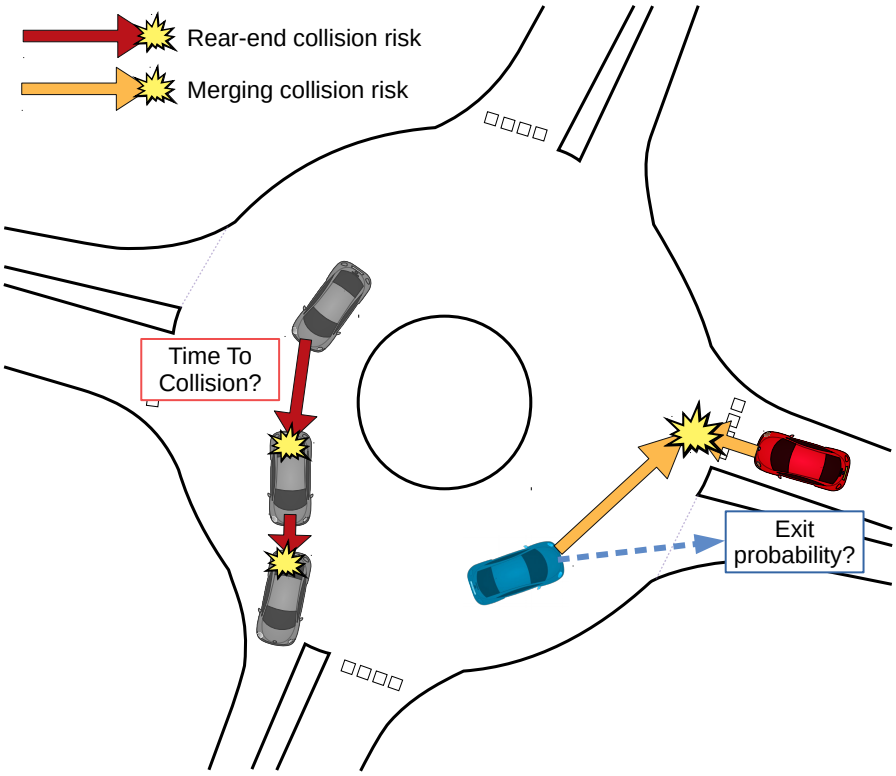


Figure 5.11: Types of Conflicts and Collision Risks in Roundabouts

sequential, i.e., rear-end conflicts, for penetration rates of above 50%, lower penetrations rates show mixed effects on traffic. In low penetration rates, [185] and [186] described an increased number of conflicts, and [187] indicates that CAVs are disproportionately subject to rear-end conflicts caused by a rear human-driven vehicle behind a CAV.

Generally, as illustrated by Figure 5.11, roundabouts are subjects to two types of conflicts:

- Sequential conflicts, i.e., rear-end collision risk, for vehicles in a car-following situation in the roundabout, as shown in dark red in the figure. In turn, metrics to assess the time to a collision are required to estimate rear-end collision risk.
- Merging conflicts, e.g., (i) when a vehicle is changing lane within the roundabout, (ii) when vehicles in two different lanes try to exit the roundabout at the same time through the same exit, and (iii) when a vehicle enters the roundabout despite incoming traffic, as shown by the light orange arrows in the figure. In turn, metrics to assess the probability of vehicles to change lane or exit the roundabout are required. In this work, we focus on the exit probability rather than the lane change probability, which has been considered in existing works such as [63].

As such, both metrics to assess the risk for rear-end conflicts as well as the probability of exit of a vehicle at a given entry are required to accurately estimate the risk in the roundabout. Namely, if the computation of the rear-end conflict risk is highly accurate but does not take the probability of a vehicle to exit the roundabout,

the risk of merging conflicts remains. On the other hand, if the exit probability is computed accurately but not the risk of rear-end collisions, car following situations in the roundabout are risky.

In turn, risk metrics which are based on rear-end and merging conflicts are pertinent in the case of roundabouts, to assess the risk for CAVs of crossing a roundabout at a given time. Generally, assessing the level of risk in a specific driving situation is complex, as a situation might be interpreted differently by distinct vehicles. A situation which is considered as risky by a passenger might be considered safe for another, as studied in [188]. Moreover, in traffic rules and conventions themselves, situations might be safe in a country, but risky in another, because it is unusual or illegal there. For example, overtaking on the right side is permitted on highways in California, but prohibited in most European countries. As such, risk knowledge should be defined which can adapt to some extent to the legal context and the preferences of passengers.

In [56], we performed a generic study to infer the most risky areas in the MoST simulated city of Monaco, as introduced in Section 5.1. Namely, we report each merging, rear-end, and diverging conflict in the simulation to form clusters of high concentration of conflict points.

In this section, we specifically focus on a more in-depth study of driving risk in roundabouts related to rear-end conflicts, while taking the probability of vehicles to exit the roundabout into account. To objectively measure rear-end conflicts, we use the popular Time To Collision (TTC) metric. TTC is a measure of the time remaining before a rear-end collision in a car following situation. Namely, let us consider a vehicle v_{rear} following a vehicle v_{front}, such that at time t the instantaneous velocity of v_{rear} is greater than that of v_{front}, i.e., if both velocities remain constant, a collision will occur. The TTC at time t is the time remaining before a collision, if both vehicles were to keep a constant velocity.

In turn, a common approach to detect rear-end conflict is to set a threshold TTC_{th} for TTC values, such that when $TTC(v_{rear}, v_{front}) < TTC_{th}$, a rear-end conflict is notified. Different values of TTC thresholds may be selected, depending on the situation. For example, [189] suggested critical TTC thresholds of 1 to 1.5 seconds. Later, research based on crash statistics found the average TTC value at the time of hard breaking before a crash to be located between 1.1 and 1.4 seconds, in [190].

To begin with, we describe a means of computing TTC values in the circular roads of a roundabout, which we apply to real vehicle track data extracted from the *RounD* drone-captured dataset [191]. In turn, we define driving risk knowledge in roundabouts through three complementary contributions:

1. On the one hand, we train a knowledge model to assess the probability of vehicles to exit a roundabout in the next available exit. This knowledge can be used to (i) further refine the TTC computation, by ignoring cases where a vehicle is likely to exit before a rear-end conflict, or (ii) on its own, e.g., to let vehicles which enter a roundabout assess whether an incoming vehicle will exit or drive in front of them, as detailed in Chapter 6.

2. Based on TTC values computed in the considered roundabout, as well as the probability of vehicles to exit the roundabout before encountering a conflict, we define and compare several TTC-based rear-end conflict risk metrics for roundabouts.

3. Finally, we investigate the relationship between the risk metrics which we described and the seasonality, i.e., rate of variation of TTC values over time in a roundabout, in order to potentially define an estimation of roundabout driving risk knowledge by looking at the variation of TTC values.

5.2.1 Roundabout Time to Collision Extraction

In this section, we introduce the RounD dataset and describe a process to compute TTC values for extracted pairs of vehicles. RounD is a dataset of drone-captured vehicle tracks, provided by Krajewski *et al.* [191]. It contains real vehicle tracks extracted from three German roundabouts. The tracks feature a positioning error which is typically below 10cm. The dataset is divided into 24 recordings of 15 minutes of recorded vehicle tracks, among which 22 were extracted from a single roundabout, which is illustrated by Figure 5.12. Because of its extensive amount of tracks data, we use this roundabout to compute TTC values between pairs of vehicles. The roundabout has the identifier '0' among the three RounD roundabouts. It has two lanes in its circular part and in its four entries. What is more, it features four single-lane exits.

We use tracks data from the considered roundabout to compute TTC values. As the TTC involves rear-end conflicts, i.e., a front and a rear vehicle, a first step is to define a procedure $front(v)$ to detect the vehicle which is directly in front of a vehicle v. As roundabouts feature circular lanes, which, in the case of the considered roundabout, are not clearly separated by ground markings, a specific procedure is required.

Front Vehicle Detection

We define a procedure to detect the front vehicle of a vehicle v in the circular part of a roundabout. First, we add reference points to the considered roundabout in order to define discrete cells in the roundabout, which we use to model the position of each vehicle. Namely, the circular part of the roundabout is divided into several virtual lanes, each of width W_L. In turn, each virtual lane is itself divided into N_{slices} slices, each featuring an arc measure of $\frac{2\pi}{N_{slices}}$ rad. Using these reference points, we define a (L, S) discrete coordinate system for vehicles, with L and S, respectively, the lane and the slice of the lane in which the center point of the considered vehicle is located. To avoid the eventuality that two vehicles could share the same coordinate in the roundabout, W_L and N_{slices} are chosen such that only one vehicle can occupy a cell at a given time. Considering the size of passenger vehicles in the dataset, we set $W_L = 2.25m$ and $N_{slices} = 30$.

Figure 5.13 illustrates the virtual lanes and their slices, which were defined for the

Figure 5.12: Layout of the Considered Roundabout [191]

considered roundabout using light blue-colored lines. The annotated dark red boxes represent vehicles with their identifier in the considered track file. As we defined the lanes and slices such that the center point of two vehicles cannot be located in the same cell, we can use the discrete position of each vehicle to determine the front vehicle of v. As described in Algorithm 3, the front vehicle of v is the vehicle which intersects a slice in front of the slice of v. If no vehicle has been detected through the $\frac{N_{slices}}{2}$ slices in front of v, no front vehicle is notified.

TTC Computation

After the front vehicle $front(v)$ of v was identified, and provided v has a greater velocity than $front(v)$, a TTC value can be computed for the $(v, front(v))$ pair. In straight road segments, the TTC between two vehicles is computed as the Cartesian plane distance between the front of the rear vehicle and the rear of the front vehicle, divided by the velocity difference between the rear and the front vehicle. However, this definition does not apply in the circular part of roundabouts, as vehicles follow the curve of the roundabout.

To compute TTC in the circular part of a roundabout, we compute the distance between both considered vehicles as the length of the circle arc between the vehicles, centered on the center point of the roundabout. Figure 5.14a illustrates the TTC computation process which we define for roundabouts. Specifically, the equation for

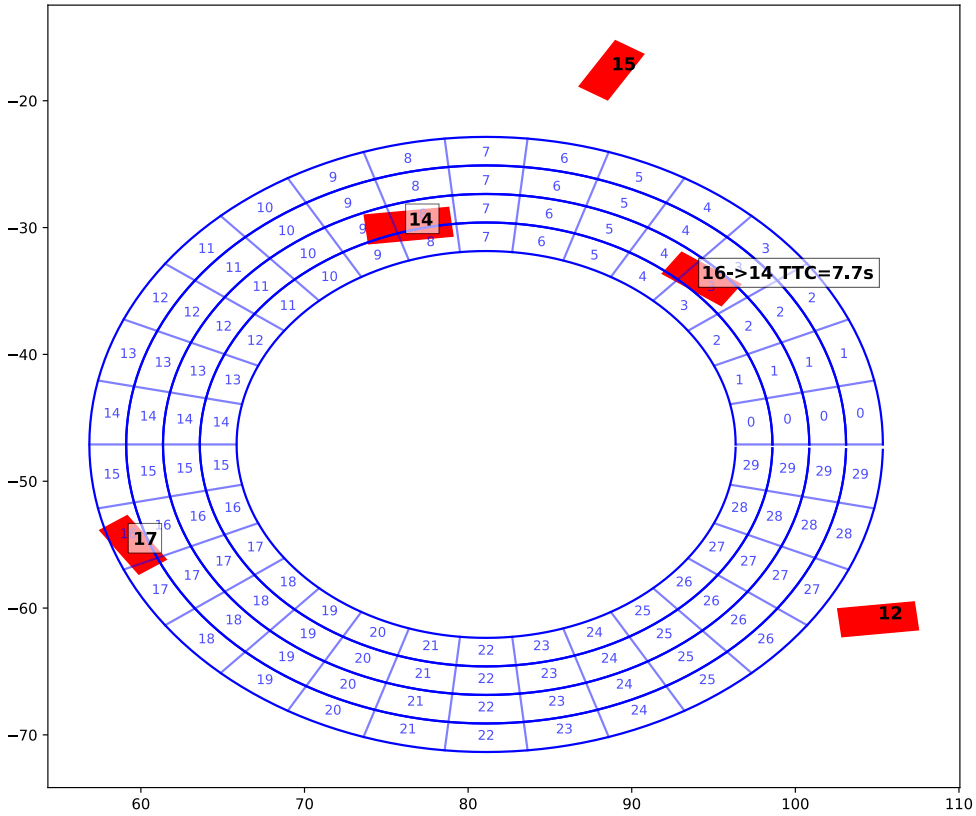
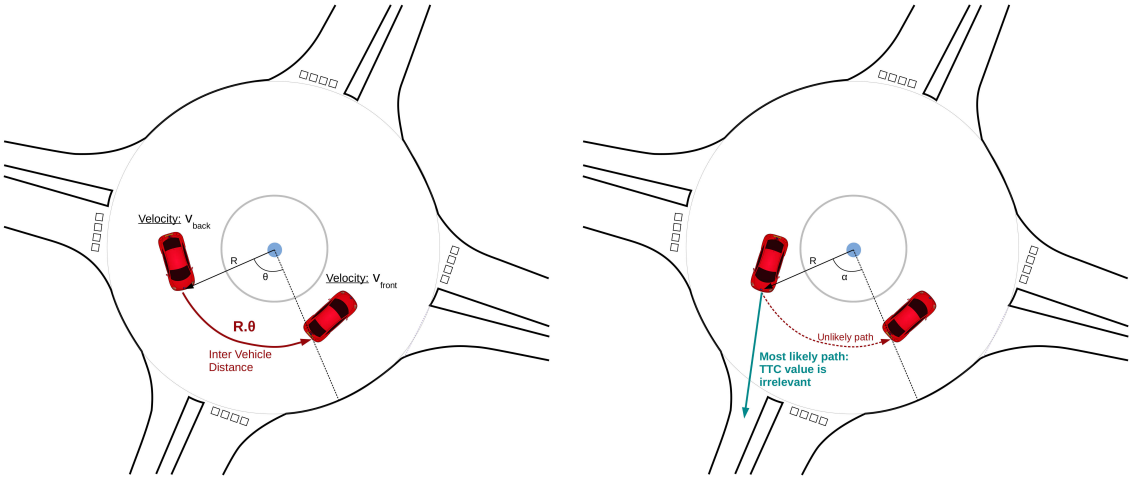


Figure 5.13: Front Vehicle Detection and TTC Computation



(a) TTC Computation in a Circular Roundabout Area

(b) Limits of TTC Computation in Roundabouts

Figure 5.14: Roundabout TTC Computation and Challenges

Algorithm 3 Detection of a Front Vehicle in RounD Tracks

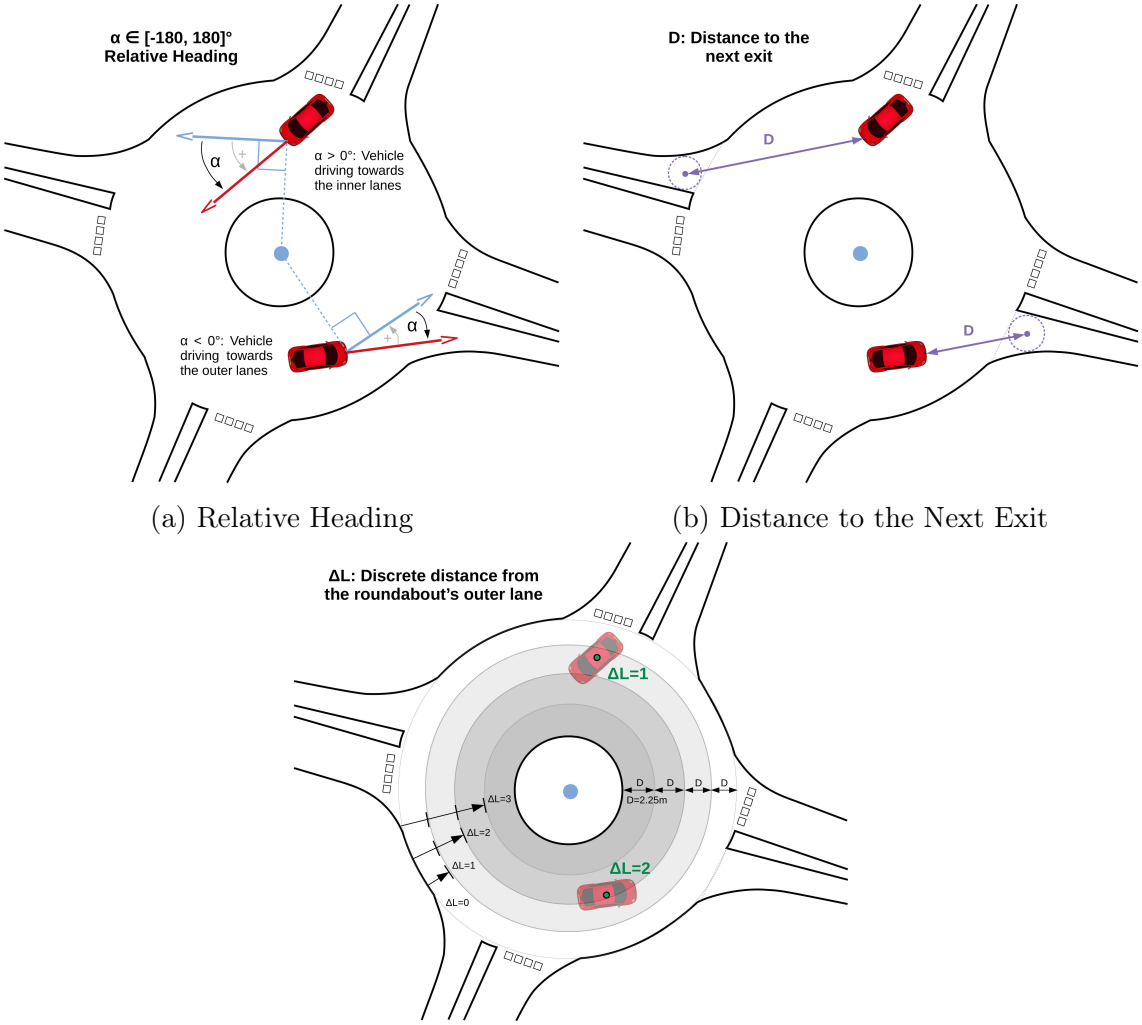
– Let $front\_slice(L, S)$ a function which returns the adjacent slice of S in lane L in anti-clockwise direction.

– Let $intersects(S, v)$ a function which returns whether v has a non-null intersection with slice S.

– Let V the set of vehicles currently in the roundabout area.

procedure front(v)

Let N_{slices} the number of slices of each lane.
Let c the centroid of vehicle v.
Let L the ID of the lane containing c.
Let S the slice of ID i of Lane L, where c is located.
Let $i \leftarrow 0$.
Let $S\_iterate \leftarrow S$.
Let $front\_v \leftarrow null$.
while $i < \frac{N_{slices}}{2}$ **and** $front\_v = null$ **do**
    $S\_iterate \leftarrow front\_slice(L, S\_iterate)$
    for all $w \in V \setminus \{v\}$ **do**
        if $intersects(S\_iterate, w)$ **then**
            $front\_v \leftarrow w$
            break
        end if
    end for
    $i \leftarrow i + 1$
end while
return $front\_v$



(a) Relative Heading

(b) Distance to the Next Exit

(c) Lateral Position

Figure 5.15: Inputs for Roundabout Exit Probability Assessment

computing the TTC is as follows. R, θ, V_{back} and V_{front} are defined in Figure 5.14a.

$$TTC = \frac{R \cdot \theta}{V_{back} - V_{front}}$$

5.2.2 A Model for Exit Probability Estimation

The computation of TTC values can be used to detect rear-end conflicts in a roundabout. As described in [189, 190], TTC values which are measured for a pair of vehicles under a critical TTC threshold can be considered to point out a risky situation of rear-end conflict. In these papers, the value of 1.5s as a TTC threshold has been put forward in situations of highway traffic, in which vehicles have few chances to take an exit away from the road.

Yet, the situation differs in the circular part of a roundabout, as vehicles have many opportunities to exit. In turn, if an exit is located between a rear vehicle v and a front vehicle w, v may exit the roundabout before having a chance to encounter w. As such, even if the TTC between v and w has been computed as critical following the process illustrated in Figure 5.14a, there might be no real risk if v intends to exit the roundabout. Figure 5.14b illustrates this challenge.

In turn, the fact that vehicles may exit the roundabout before encountering rear-end collision risk should be accounted for to avoid 'false positives' of considering critical TTC values which are not risky due to rear vehicles exiting the roundabout. In [63], a lane change probability estimation model was computed to weight the collision risk with the probability of lane change. Similarly, we define and train a model to assess the probability of a vehicle to exit the roundabout at the next available exit, in order to weigh critical TTC values.

Model Definition

In this section, we define and train a supervised ML model to assess the probability of a vehicle v to exit the roundabout at the next available exit, directly in front of it, based on its current position in the roundabout. Namely, as illustrated by Figure 5.15, we select the following input to estimate the probability of exit of the vehicle:

- The heading of the vehicle $\alpha \in [-180, 180]deg$, relatively to the curvature of the roundabout:
  - If $\alpha = 0$, the vehicle is following the curve of the roundabout.
  - If $\alpha < 0$, the vehicle is driving towards the outer lanes of the roundabout.
  - If $\alpha > 0$, the vehicle is driving towards the inner lanes of the roundabout.
- The straight-line distance D between the front bumper of the vehicle and the next available exit.

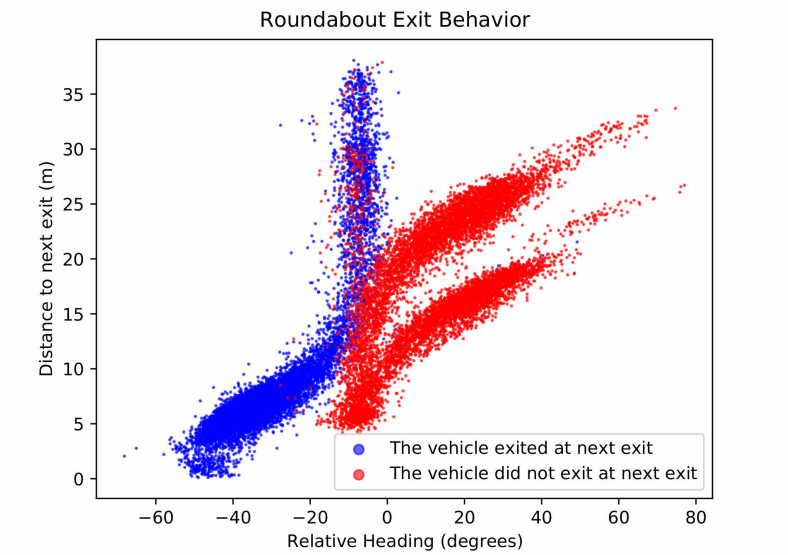


Figure 5.16: Roundabout Exit Data for Vehicles in the Outermost Lane

- The discrete lateral position of the vehicle in the roundabout, i.e., the identifier ΔL of the virtual lane in which the vehicle is located, starting from $\Delta L = 0$ for the outermost lane of the roundabout.

We select α as an input feature of the model as we anticipate it to show a strong correlation with the intention of the driver whether to exit at the next exit. What is more, we add L and D to indicate information about the position of the vehicle relatively to the exit. Finally, these three input items can realistically be sensed by connected vehicles in real situations, making the model usable in practice.

Finally, we select *logistic regression* as the supervised ML algorithm to train the model. It is adapted as (i) the class to estimate is binary, i.e., exit or no exit, and (ii) in addition to a classification of whether the vehicle will exit or not based on the $(\alpha, D, \Delta L)$ input parameters, it outputs an estimation of the probability of the vehicle to exit at the next exit. The output probability is a key target to weigh the risk of rear-end conflict. Generally, in real applications, it allows to finely adapt the behavior of CAVs based on the level of certainty that a vehicle will exit or not.

Training Data Collection

To train the logistic regression supervised model and predict the probability of a vehicle to exit, a large training set is required, which associates entries of $(\alpha, D, \Delta L)$ with whether the vehicle did exit. In turn, the vehicle tracks of the 22 recordings which are available for the considered roundabout are replayed. For each frame of each recording, if the center point of a vehicle is present in the circular part of the roundabout, the $(\alpha, D, \Delta L)$ input is collected, and later associated with whether the vehicle did exit at the next available exit. Overall, 2269561 training entries were extracted accordingly.

Figure 5.16 represents a part of the obtained training entries. It considers the entries which were collected for vehicles driving in the outermost virtual lane of the

roundabout, as shown in Figure 5.13. The blue-colored points show vehicles which did exit the roundabout at the next available exit, whereas the red-colored points show the vehicles which stayed in the roundabout. A pattern can be observed in the data. Vehicles which are close to the next exit are likely to exit if they are clearly headed towards the outer part of the roundabout. On the contrary, vehicles which are headed towards the inner part of the roundabout are more likely to stay in the roundabout.

Model Evaluation

The extracted training entries are randomly shuffled. Then, they are separated into a training set, representing 80% of the entries, and a validation set containing the remaining 20%. Using the training set, a logistic regression ML knowledge model is trained. In turn, the predictions of the model on the entries of the validation set are compared with the ground truth, i.e., the real observed exit patterns of vehicles. As the model produces the probability p of exit as an output, an entry is predicted as an exit iif $p > 0.5$. The accuracy of the predictions of the trained model on the validation entries reaches 91% of correct predictions. Figure 5.17 illustrates the probabilities of exit which were predicted by the model for vehicles which are driving on the outermost lane of the considered roundabout.

First, we notice that the more vehicles are oriented towards the outside of the roundabout, the more they are likely to exit at the next exit, and vice versa. What is more, we notice that the closer vehicles get to the exit, the more their heading must be oriented towards the outside of the roundabout to predict that they will exit the roundabout. For example, the probability of a vehicle which is following the curve of the roundabout, i.e., with a relative heading of 0deg, to exit at the next exit if it is 20m away is around 30%. On the contrary, if the vehicle is 5m away from the exit with the same relative heading, its probability of exit is close to zero. This can be interpreted by the fact that, in practice, exiting the roundabout at the next exit when it is only a few meters away while following the curve of the roundabout requires a sudden turn towards the exit at the last moment, which is rarely observed on roundabouts.

5.2.3 Roundabout Rear-End Conflict Risk Metrics

Based on the procedure to compute TTC values for pairs of vehicles in the considered roundabout, as well as the model to estimate the probability of a potential rear vehicle to exit before encountering risk with a front vehicle, as shown in Figure 5.14b, we provide a set of three incrementally sophisticated risk metrics to estimate the roundabout-wide level of rear-end conflict risk at a given time period.

We consider each of the 22 available recordings for the considered roundabout in the RounD dataset. For each recording:

- We train an exit probability estimation model using the training entries from the 21 other recordings, so that the model is not applied on its training data.

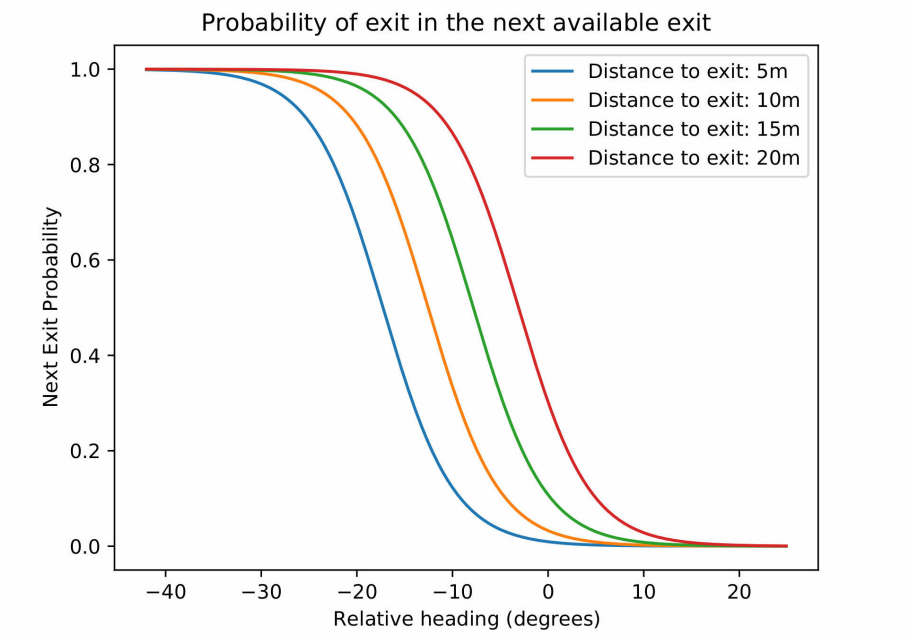


Figure 5.17: Model Estimation of the Probability of Exiting the Roundabout for Vehicles in the Outermost Lane

Each of the 22 produced models features a prediction accuracy which is greater than 90%.

- Then, each frame of the recording is replayed. For each frame, the TTC values of each pair of vehicles in the circular part of the roundabout are computed.

Both aspects are used to produce three incrementally sophisticated roundabout-wide driving risk metrics for various levels of risk, which we name (i) **number of rear-end conflicts**, (ii) **accumulated TET**, and (iii) **probability-weighted accumulated TET**.

Preliminary Considerations

Namely, as introduced in Section 5.2, in order to detect situations of risk, TTC values need to be compared with a critical TTC threshold under which the situation is flagged as risky. Different values of TTC thresholds can be chosen depending on the context or preferences of each passenger or CAV. In [192], human drivers have been given the option to change their preferred value of TTC threshold before an alarm rings in the vehicle, to adapt it to their driving preferences. Alternatively, approaches to dynamically update TTC thresholds based on an algorithm to analyze the behavior of drivers have been considered, as in [61]. In turn, we define risk metrics based on a choice of a value of TTC threshold, so that they can be used according to any driving preference.

Namely, we define risk metrics which can be computed for any value of TTC threshold TTC_{th}. When computing the risk metrics on real RounD recordings, we use the values of $TTC_{th} \in [1, 2, 3, 4, 5, 6]$ seconds. Values above 5 seconds

are conservative thresholds, which have been considered to trigger early collision avoidance systems in [189]. Values between 1 and 2 seconds are critical collision risk thresholds, as described in [189] and [190]. Finally, a TTC threshold of 3 to 4 seconds is a moderately conservative threshold, representing moderate rear-end collision risk.

Definition of Three Driving Risk Metrics

Then, for each recording, we already computed the TTC values of pairs of vehicles at each frame, as well as a model to assess the probability of a vehicle to exit the roundabout at the next available exit. Based on these items, we define and compute the following three risk metrics for each value of TTC threshold $TTC_{th} \in [1, 2, 3, 4, 5, 6]$ seconds:

1. We define the **number of rear-end conflicts** metric as the number of rear-end conflicts in the recording, i.e., the number of situations in which the TTC for a pair of vehicles has been lower than TTC_{th}. Only one rear-end conflict per second is counted for a single pair of vehicles.

2. We define the **accumulated TET** as the accumulated time which was spent under the critical TTC threshold TTC_{th} by pairs of vehicles in the recording. This is equal to the accumulated Time-Exposed TTC (TET) of vehicles of the recording, as initially described in [193].

3. Finally, we contribute the **probability-weighted accumulated TET** metric. It is a novel metric which weights the accumulated TET using the probability of vehicles to exit the roundabout. Namely, the probability of the rear vehicle to exit before encountering the risk is taken into account, using the training exit probability estimation model.

Accumulated TET The **accumulated TET** risk metric is computed as follows:

- At frame 0, the risk metric R_{TET} is initialized, $R_{TET} \leftarrow 0$.

- For each frame and each pair of vehicles (v_1, v_2), if $TTC(v_1, v_2) < TTC_{th}$, R_{TET} is updated, as,

  $R_{TET} \leftarrow R_{TET} + \frac{1}{F}$, with $F = 25Hz$ the frame rate.

Yet, the first and the second risk metrics include situations in which a rear vehicle and a front vehicle featured a critical TTC value, but the rear vehicle was clearly headed towards an exit, as illustrated in Figure 5.14b. In turn, 'false positives' of risk are included in the computation. To address this limitation, the third risk metric R^p_{TET} takes the probability of exit of the rear vehicle into account, in case an exit is located between the rear and the front vehicle.

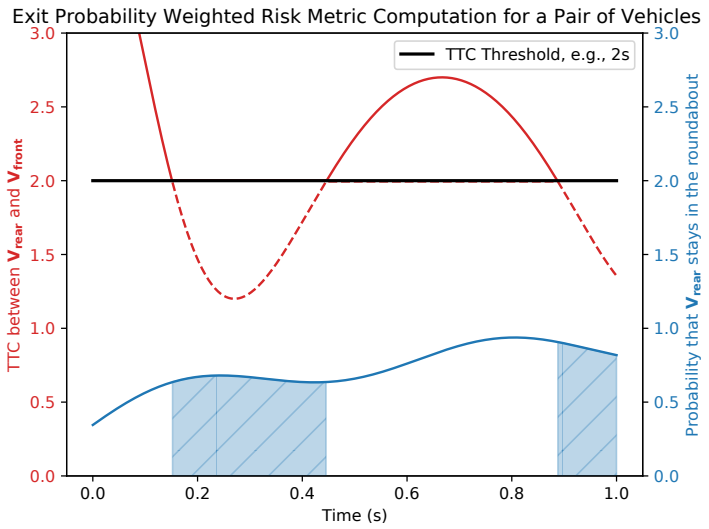


Figure 5.18: Example of the Computation of the Exit Probability Weighted Risk Metric

Probability-weighted Accumulated TET In turn, we contribute a novel driving risk metric R^p_{TET}, which weights the time spent under a critical TTC threshold by the probability of the rear vehicle to exit the roundabout before encountering the collision risk. The pseudocode of the algorithm to compute this **probability-weighted accumulated TET** metric is listed in Algorithm 4, in the *compute_recording_risk* procedure.

Figure 5.18 illustrates the computation of R^p_{TET} for a single pair of vehicles (v_{rear}, v_{front}), in a car following situation for 1 second. When $TTC(v_{rear}, v_{front}) < TTC_{th}$, the time spent under critical TTC threshold, i.e., $TTC_{th} = 2s$ in this example, is counted. Yet, if an exit is located between v_{rear} and v_{front}, the time spent under critical threshold is integrated by the probability of v_{rear} to stay in the roundabout, i.e., actually produce a risk of rear-end conflict.

Namely, Figure 5.18 graphically represents the computation of the R^p_{TET} **probability-weighted accumulated TET** risk metric for two vehicles in a roundabout. As shown in the dashed dark red line, the time in which in TTC for the two vehicles was below the critical TTC threshold is counted. What is more, as shown by the light blue line and filled areas in the bottom of the figure, the time spent under the critical threshold is weighted by the probability of the TTC threshold to be associated with rear-end collision risk, i.e., the probability of the rear vehicle not to exit before encountering the risk, if an exit if located between the rear and the front vehicle.

Algorithm 4 Computation of the Probability-Weighted Accumulated TET Roundabout Risk Metric

$framerate \leftarrow 25Hz$ ▷ The frame rate of the considered recording
$TTC_{th} \in [1, 2, 3, 4, 5, 6]$ ▷ The used critical TTC Threshold
procedure $risk\_probability(v_1, v_2)$
    if $no\_exit\_between(v_1, v_2)$ **then**
        return 1
    else
        return $1 - P_{exit}(v_1)$ ▷ The probability of v_1 not to exit at the next available exit
    end if
end procedure

procedure $compute\_recording\_risk(recording)$
    $R^p_{TET} \leftarrow 0$
    for each frame f of $recording$ (the vehicle tracks recording) **do**
        for each vehicle pair (v_1, v_2) in the circular part of the roundabout **do**
            if $TTC(v_1, v_2) < TTC_{th}$ **then**
                $R^p_{TET} \leftarrow R^p_{TET} + \frac{risk\_probability(v_1, v_2)}{framerate}$
            end if
        end for
    end for
    return R^p_{TET}
end procedure

After each recording was replayed and each risk metric computed for $TTC_{th} \in [1, 2, 3, 4, 5, 6]s$, the risk metrics are normalized by the duration of the recording. In turn, the obtained values provide a level of risk per time unit in the roundabout throughout the length of the recording. As such, road users may be provided with real-time knowledge of the current level of risk in a roundabout, depending on their preferred TTC_{th} TTC threshold value.

5.2.4 The Variation of TTC as a Complementary Risk Indicator

In this section, we evaluate the evolution of roundabout risk through time, using the three roundabout-wide risk metrics which we defined in Section 5.2.3. Specifically, we investigate the relationship between the level of driving risk in a roundabout and the seasonality of TTC values which occur in its circular part, i.e., their level of variation. If a relationship exists between these two quantities, a supplementary risk indicator can be inferred for roundabout driving risk, which is the variation of TTC values within the roundabout. To compute the variation of TTC values in a roundabout, we use the normalized metric of the Relative Standard Deviation (RSD) of TTC values, otherwise called coefficient of variation.

What is more, in a second part, we consider the impact of positioning error on the assessment of risk through the variation of TTC in a roundabout. Namely, we consider two cases. First, we describe the computation of the relationship between the RSD of TTC values and the roundabout risk metrics which we defined

in Section 5.2.3 without positioning error. In turn, we add positioning error in the computation of TTC values and evaluate the obtained results.

Computation Setup

To evaluate the relationship between TTC variation and the risk metrics defined in Section 5.2.3, we compute both quantities for a set of vehicle tracks data. To begin with, each of the 22 RounD recordings of the considered roundabout, which are each about 15 minutes long, are sliced into a set of subrecordings of a duration of T seconds.

For each subrecording, the risk metrics are computed for various TTC thresholds $TTC_{th} \in [1, 2, 3, 4, 5, 6]s$, as described in Section 5.2.3. What is more, the RSD of TTC values in the subrecording is computed. Namely, for each frame, the TTC values which fall in the range of $[0, TTC_{max}]$ seconds are averaged to produce a roundabout-wide value. The RSD of these roundabout-wide values is then computed.

Impact of Positioning Error

In real situations and implementations, CAVs suffer of some extent of positioning error when estimating their position, and in turn that of other sensed vehicles. The error which comes from Global Navigation Satellite Systems (GNSS) can be lowered through sensor fusion and Real Time Kinematic (RTK) algorithms. As considered in [194], current systems can position the ego vehicle with a centimeter-level precision.

As such, we introduce various levels of uncorrelated positioning error when computing TTC values, in order to replicate the precision loss due to measurement when sensing the position of the ego vehicle or that of a sensed vehicle:

- We sample the positioning errors from a normal distribution $\mathcal{N}(0, \sigma^2)$ of mean 0.0m, and standard deviation σ, which represents the size of the error.
- When a TTC value is computed between two vehicles, positioning error is added to both their positions for the computation.
- The process of roundabout-wide risk metrics and TTC variation computation described in Section 5.2.4 is repeated for (i) No positioning error, (ii) $\sigma = 0.3m$, (iii) $\sigma = 0.5m$, (iv) $\sigma = 1m$ and (v) $\sigma = 5m$.

Results

A linear relationship can be observed between the RSD of the TTC values and the roundabout-wide risk metrics, which were jointly computed for each subrecording. The correlation is most significant after tuning the subrecordings length to $T = 450s$ and the maximal considered TTC value to $TTC_{max} = 7.5s$.

Figure 5.19a shows scatter plots representing the linear relationship between the RSD of TTC values, and the **probability weighted accumulated TET** risk metric as defined in Section 5.2.3, computed for a TTC threshold of $TTC_{th} = 2s$. Graphs obtained using the two other risk metrics which we defined in Section 5.2.3 are available in our publication associated with this work item in [21]. The linear regression fit is illustrated for each graph, with the 95% prediction and confidence intervals shown in gray areas. At a RSD value of 0.55, an outlier corresponds to a recording which involves three vehicles parked in the innermost lane of the roundabout. This increased the number of rear-end conflicts and RSD of TTC values.

In turn, Figure 5.19b illustrates the impact of positioning error on the strength of the linear relationship between the RSD of TTC values and the **probability weighted accumulated TET** roundabout-wide risk metric

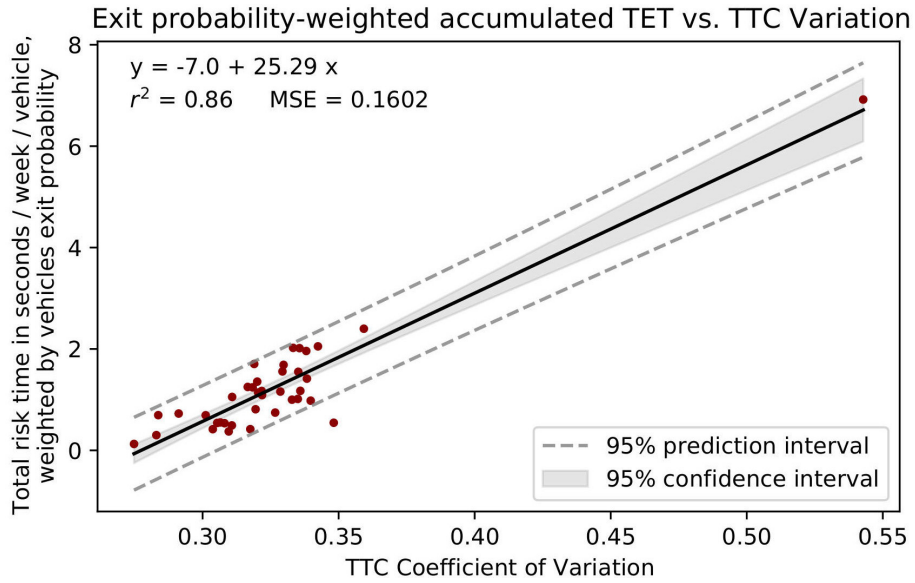
On the one hand, the light blue-colored curves represent the r^2 coefficient of determination of the linear relationship obtained for different scales of positioning errors. The *errorless* r^2 is the r^2 which was obtained for the TTC values computed from the original track data, without positioning error. As such, strong correlation can be obtained between the 'probability-weighted accumulated TET' metric and the variation of TTC values up to a TTC threshold of $TTC_{th} = 3.0$ seconds. What is more, the relationship is robust to increasing positioning errors.

On the other hand, Figure 5.19b illustrates the impact of the positioning error which was introduced in the TTC computation, on the ability of the obtained linear fit to estimate real values of roundabout-wide risk metrics. Namely, for each level σ of positioning error, the Root-Mean-Square Error (RMSE) of the obtained linear fit on the scatter of *ground truth* $(RSD, risk\ metric\ value)$ points is computed. In turn, it is divided by the RMSE obtained using the linear fit computed without positioning error. Each bar of the graphs of Figure 5.19b, with various shades of green and orange corresponding to a level of positioning error σ, represents the relative increase in the RMSE error of the linear fit computed with σ positioning error, to estimate the roundabout-wide risk metrics. 95% confidence intervals are provided after running 20 simulations with different positioning error samplings for each value of σ.

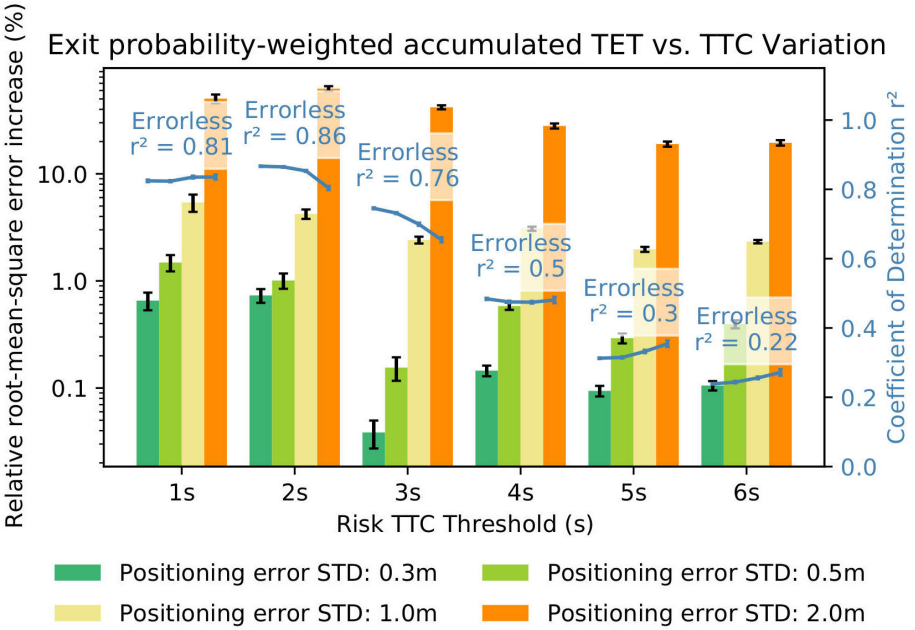
In turn, we observe that the **probability-weighted accumulated TET** metric estimation through TTC variation is robust to positioning errors. Namely, positioning errors of up to $\sigma = 1m$, which can be considered conservative using modern RTK approaches as described in [194], cause an RMSE increase below 10%.

5.3 Discussion

In this chapter, we investigated the factors of risk for CAVs in roundabouts, focusing on the specific case of rear-end and merging conflicts. In turn, after performing an initial analysis to show the need for risk knowledge, we contributed (i) a ML model for the estimation of the exit probability of vehicles, (ii) a novel roundabout-wide rear-end conflict based risk metric, which takes the probability of vehicles to exit the roundabout into account, and (iii) a complementary algorithm to extract an



(a) Scatter plot of TTC Variation and 'probability-weighted accumulated TET', for a TTC Threshold of 2 seconds



(b) Impact of Positioning Error on TTC Variation-based Risk Prediction

Figure 5.19: Linear Relationship between TTC Variation and the 'probability-weighted accumulated TET' Risk Metric

indicator of the aforementioned risk metric based on the variation of time to collision values in a roundabout.

Yet, the definition of the rear-end risk metric, as well as the model to estimate the probability of a vehicle to exit at the next available exit, has been contributed and validated using track data extracted from a unique roundabout from the RounD dataset. In turn, what is the behavior of the risk metrics and knowledge models we defined on other roundabouts? Namely, how accurate is the exit probability model trained on this roundabout, when it is applied and used in other roundabouts?

What is more, in this chapter, we defined roundabout risk knowledge with the aim to evaluate the VKN framework which we defined in Chapter 4 on a realistic instance of vehicular knowledge. The roundabout exit probability model must be semantically described in order to be distributed in the vehicular networks through VKN. Yet, how to describe the exit probability model such that it can be applied by any vehicle? What are the factors which can be used to represent the context of training of the model, based on the roundabout it has been trained on?

As such, Chapter 6 uses the roundabout exit probability knowledge model which we defined in this chapter, to investigate its semantic description and its context-aware distribution in vehicular networks using the VKN framework.

Chapter 6

Evaluation of Roundabout Knowledge Distribution through VKN

In this Chapter, we use the roundabout exit probability model defined in Chapter 5 as a realistic application to implement and evaluate VKN. The evaluation considers the networking of roundabout exit probability knowledge on multiple roundabouts, which do not all feature trained exit probability models. As such, the context of training of exit probability models must be used to support the knowledge distribution of exit probabilities in other roundabouts with similar contexts.

Generally, in this chapter, we aim to define a use case of exit probability knowledge distribution to vehicles which are entering a roundabout, as illustrated by Figure 6.1. Namely, vehicles which enter a roundabout and sense an incoming vehicle request for the probability of that vehicle to exit, to decide whether to enter and safely cross the roundabout.

Yet, no exit probability model is available for the roundabout that is crossed by the vehicles. As such, entering vehicles must request for exit probability knowledge to be created using a model which was trained in a context which is similar to the context of the current roundabout, in the hypothesis that it will also be accurate in this similar context.

In turn, the following steps are required to achieve the implementation of such context-aware knowledge creation requests:

Description An analysis must be performed to identify the variables which best describe the context of training of an exit probability model, such that a model which has been trained in a specific described context will perform accurately in a similar context. In turn, conditions of similarity between two contexts must also be defined.

Storage A description of the exit probability models must be integrated to the knowledge layer of vehicular nodes. What is more, exit probability models which share the same interface but different contexts, i.e., roundabouts, of

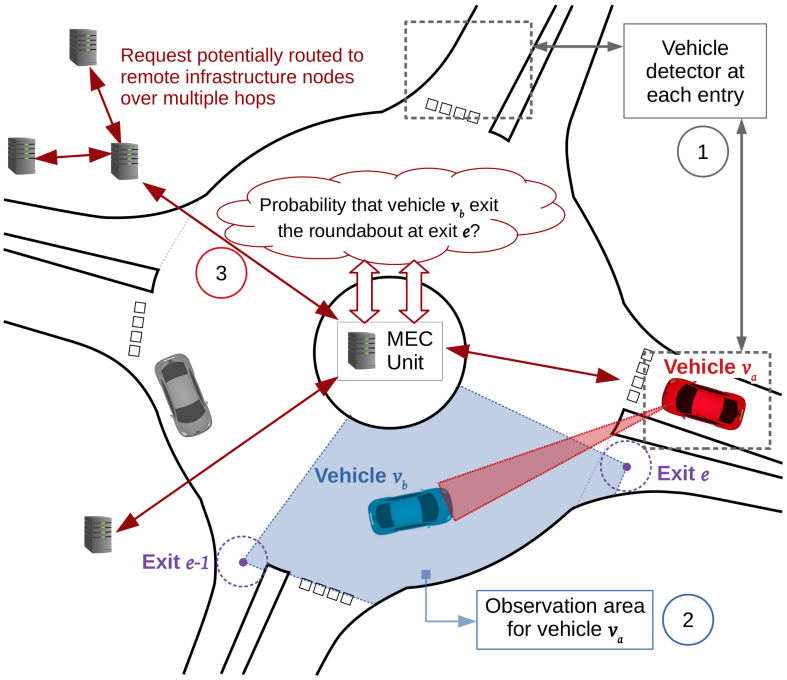


Figure 6.1: Overview of the Exit Probability Knowledge Creation Case Study

training, must share a common description, which describes the possible values that their context can take.

Dissemination Context annotations must be carried over the network when requesting for the creation of exit probability knowledge, using an unknown model, which fits for the current driving context.

To begin with, in Section 6.1, we contribute the analysis of the **knowledge description** part. Namely, we identify a context of description of exit probability models, and show that it is relevant, as models which are used in a context which match their context of training perform more accurately.

Then, in Section 6.2, based on this analysis and context definition, we perform an implementation of the knowledge storage and dissemination of VKN, which we apply to the distribution of roundabout exit probability knowledge between infrastructure units and entering vehicles, as shown in Figure 6.1.

6.1 Analysis of the Training Context of Roundabouts Exit Probability Models

In this section, we apply the exit probability model training process described in Section 5.2.2 to seven new roundabouts. Then, we evaluate the similarity between pairs of exit probability models trained on pairs of distinct roundabouts using metrics extracted from the information theory, as introduced in [195].

Based on the obtained similarity values between exit probability models trained on distinct roundabouts, we investigate whether a high similarity between two exit

probability models trained on two distinct roundabouts can be explained by similarities in the geometric of traffic characteristics of the two roundabouts. In turn, we define roundabout context description semantics, in order to assess whether two roundabouts feature a similar context which is likely to lead to similar exit probability models.

Finally, we evaluate the impact of context semantics on the accuracy of (i) exit probability model application on a distinct but similar roundabout, and (ii) exit probability model training on a distinct roundabout, assisted with training data extracted from similar roundabouts.

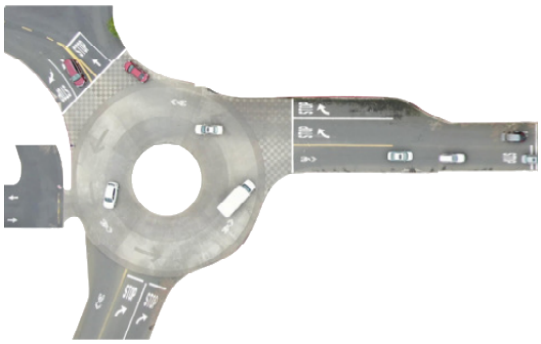
6.1.1 Considered Roundabouts

Description

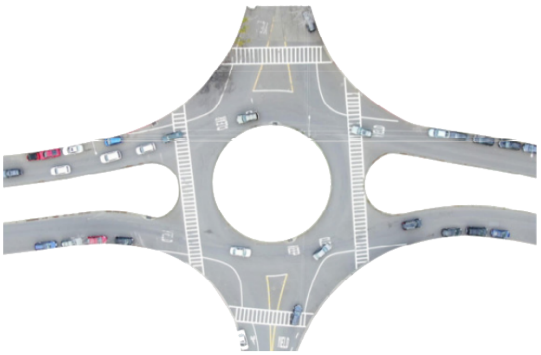
A first step to study the possibility of knowledge transfer of an exit probability model trained from one roundabout to another is to train the knowledge on a set of different roundabouts. So far, we have considered one of the three roundabouts included in the rounD dataset [191]. In turn, we train a roundabout exit probability model as described in Section 5.2.2 for (i) the two remaining roundabouts extracted from the rounD dataset, and (ii) the five roundabouts described as part of the INTERACTION dataset, as introduced in [196].

INTERACTION is a dataset of real recorded mobility tracks of vehicles in various intersections, including five roundabouts. Drone-recorded track data is available for each INTERACTION roundabout. Position, heading, and velocity data is provided at a pace of 10 traffic snapshots per second, compared with 25 for RounD. While RounD roundabouts are located in Germany in the region of Aachen, INTERACTION roundabouts are available in the USA, Germany, and China. Figure 6.2 provides an aerial picture of each of the considered roundabouts, while Table 6.1 summarizes their characteristics. The 'Entries' column refers to the number of entry legs of the roundabout. The 'Lanes' column corresponds to the number of lanes in the circular part of the roundabout, observed from the tracks data of each roundabout. The 'Radius' column corresponds to the distance in meters between the center point of the roundabout and the extremity of the innermost lane of the roundabout. Finally, the 'Width' column corresponds to the width of the circular driveable area of the roundabout from the inner extremity of the innermost lane to the outer extremity of the outermost lane.

Although the 'RounD_2' roundabout physically features four entries, the data provided by the RounD dataset does not include vehicles crossing the bottom-right entry, as shown in Figure 6.3. Namely, out of 263 tracks overall, no recorded vehicles exit the roundabout and only two vehicles enter the roundabout in this entry. As such, we flag the 'RounD_2' roundabout as practically being a 3-entry roundabout in this study.



(a) DR_USA_Roundabout_EP



(b) DR_USA_Roundabout_SR

(c) DR_USA_Roundabout_FT

(d) DR_CHN_Roundabout_LN

(e) DR_DEU_Roundabout_OF

(f) RounD_0

(g) RounD_1

(h) RounD_2

Figure 6.2: Roundabouts Considered for Exit Probability Knowledge Transfer and Context Description [196, 191]

Table 6.1: Characteristics of the Considered Roundabouts

Identifier	**Dataset**	**Country**	**Entries**	**Lanes**	**Radius**	**Width**
DR_USA_Roundabout_EP	Interaction	USA	4	1	6.75m	6.75m
DR_USA_Roundabout_SR	Interaction	USA	4	1	13.5m	4.5m
DR_USA_Roundabout_FT	Interaction	USA	7	1	9m	9m
DR_CHN_Roundabout_LN	Interaction	China	4	2	23m	9m
DR_DEU_Roundabout_OF	Interaction	Germany	3	1	8.75m	4.5m
RounD_0	RounD	Germany	4	2	15m	9m
RounD_1	RounD	Germany	4	1	8m	4.5m
RounD_2	RounD	Germany	3	1	6.75m	4.5m

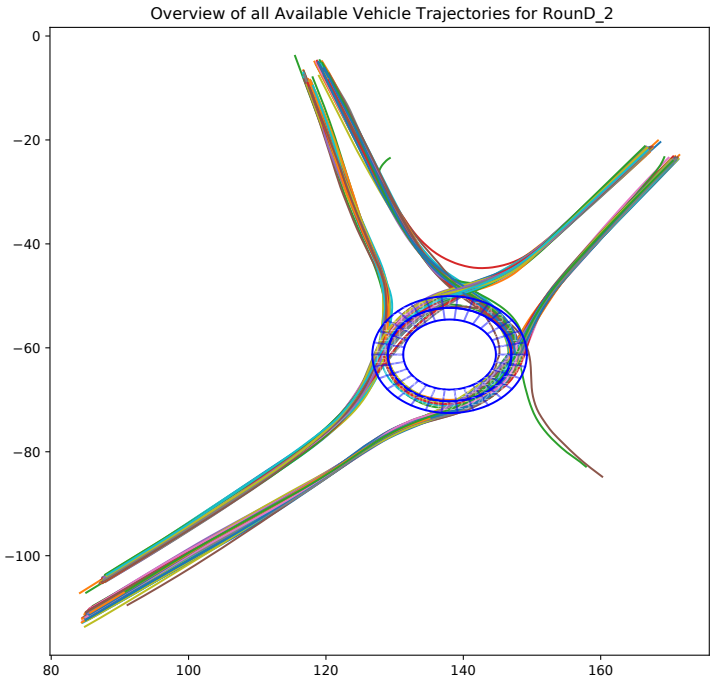


Figure 6.3: Available Vehicles Tracks for the 'RounD_2' Roundabout

Exit Probability Model Training

To study the potential for knowledge transfer between the exit probability models trained for the different roundabouts listed in Table 6.1, an exit probability model is trained for each roundabout using the process described in Chapter 5.2.2. To summarize, the vehicle tracks of each roundabout are replayed. Then, for each frame and for each vehicle in the circular part of the roundabout: (i) The relative heading, (ii) the distance to the next exit, and (iii) the lateral position in the roundabout of the vehicle are extracted and associated with whether the vehicle will or not exit the roundabout at the current next available exit. Figure 5.15 illustrates how the three $(heading, distance, lateral\ position)$ inputs are computed for a specific vehicle. Finally, a logistic regression model is trained which returns the probability of exiting the roundabout at the next available exit from the three aforementioned inputs.

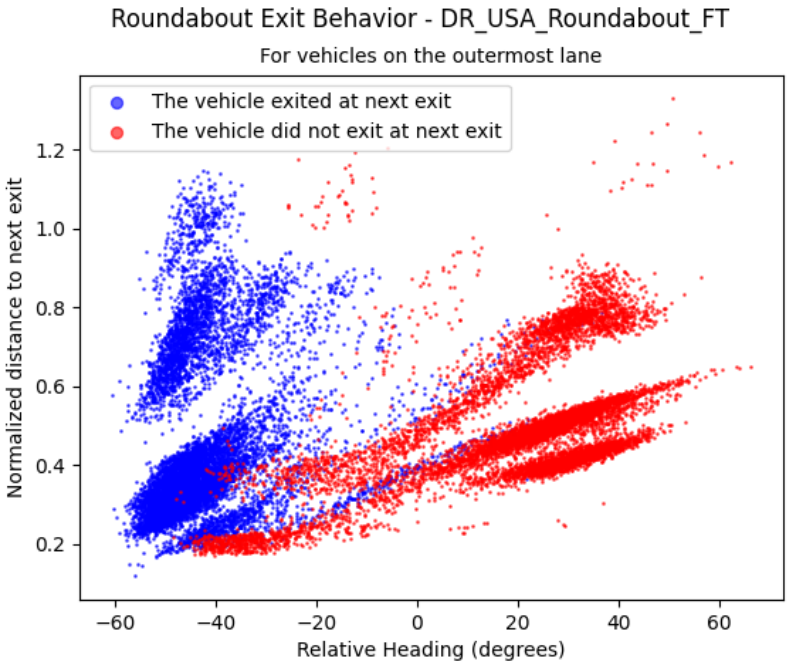
In this case, as the roundabouts listed in Table 6.1 feature different sizes and geometric characteristics, we proceed to a normalization of the $(heading, distance, lateral\ position)$ input data before training:

1. The relative heading is left as-is, as it is already a normalized value in the [-180,180] degrees range.

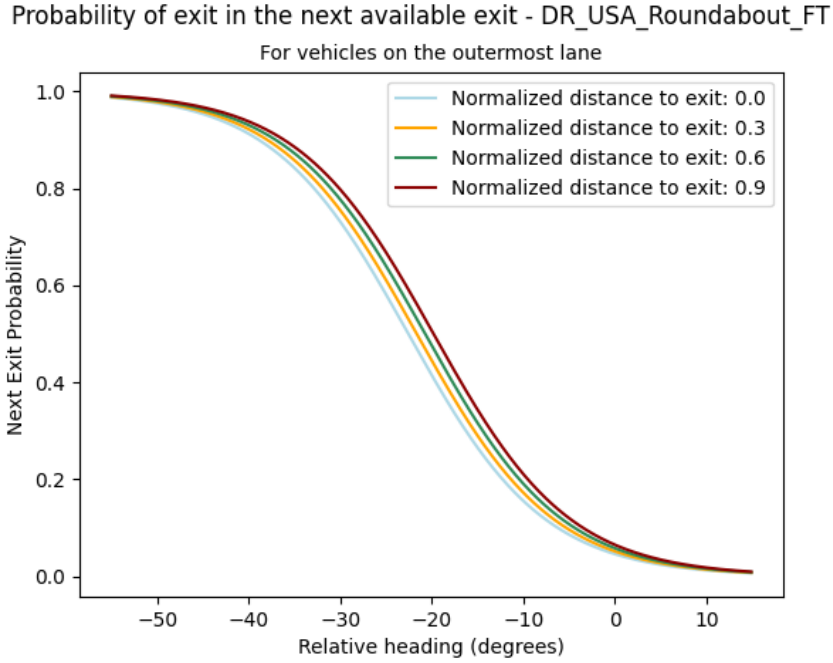
2. The distance to the next exit is divided by the distance between the next exit and the previous exit, to obtain a unitless normalized value D. For example, if a vehicle v is driving towards $exit_2$ after crossing $exit_1$, then the normalized distance to next exit is $D = \frac{distance(v, exit_2)}{distance(exit_1, exit_2)}$, with $distance(x, y)$ the straight-line distance between the x and y 2D points.

3. The lateral position in the roundabout is computed as the identifier of the virtual lane where the center point of the vehicle is currently located, starting from 0 for the outermost lane, as described in the right graph of Figure 5.15. The size of virtual lanes is fixed to a width of 2.25m, as described in Chapter 5.2.1. The roundabouts in Table 6.1 have different widths and in turn different amounts of virtual lanes. As such, the lateral position in the roundabout of a vehicle v is divided by the amount of virtual lanes in the roundabout to be normalized in the $[0, 1]$ range.

Figures 6.4a and 6.5a respectively illustrate the training data collected in the outermost lane of the `DR_USA_Roundabout_FT` and the `DR_CHN_Roundabout_LN` roundabouts of the INTERACTION dataset. After training, the exit probability models produce exit probabilities adapted to each roundabout. As an illustration, Figures 6.4b and 6.5b respectively show the probability of exit depending on the normalized distance to next exit and relative heading of a vehicle in the USA-based and Chinese roundabout. We notice different patterns for different roundabouts. For example, due to the short distance between exits in the 7-entry American roundabout, the normalized distance to the next exit has less impact on the probability of exit than in the Chinese case where exits are more spaced. In the next section, we investigate the possibility of transfer of the exit probability model knowledge trained on one roundabout listed in Table 6.1 to another.



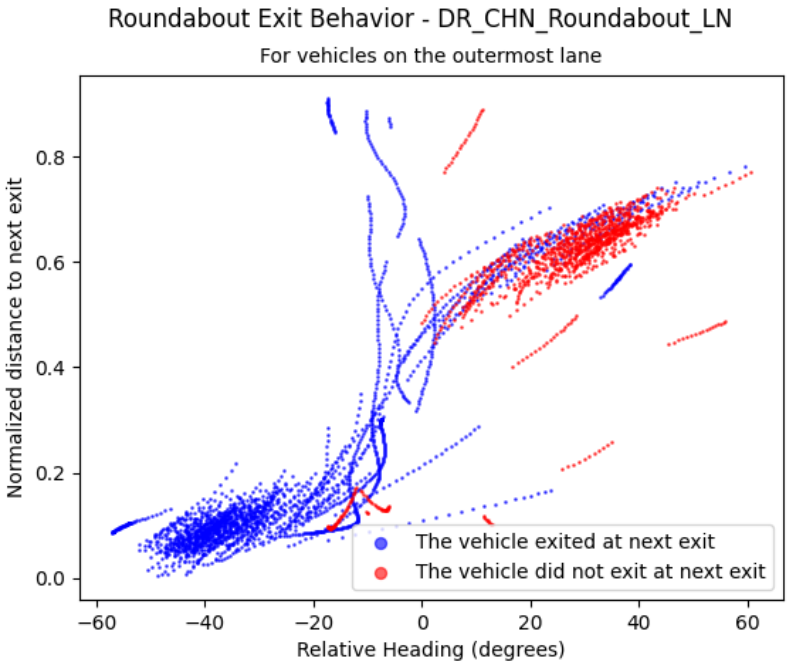
(a) Exit Pattern Scatter



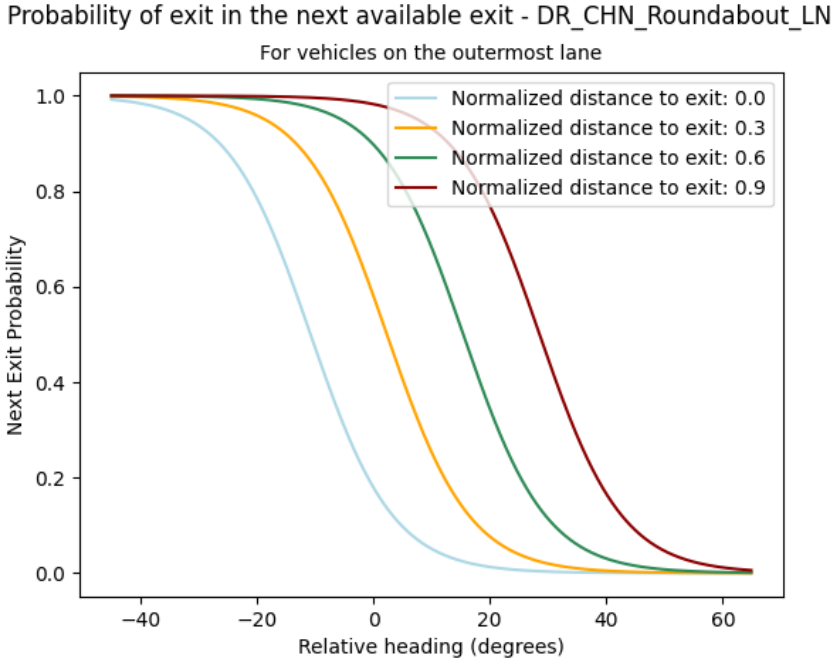
(b) Exit Probability Estimation

DR_USA_Roundabout_FT

Figure 6.4: Training Input and Trained Model Application for a USA-based Roundabout



(a) Exit Pattern Scatter



(b) Exit Probability Estimation

DR_CHN_Roundabout_LN

Figure 6.5: Training Input and Trained Model Application for a Chinese Roundabout

6.1.2 Similarity of Roundabout Exit Probability Models

We aim at studying the definition of a relevant semantic description for the context of training and use of exit probability models, and, in turn, the potential for knowledge transfer between exit probability models trained on different roundabouts. In other terms, we research to what extent a model trained using the exit input data of one roundabout can be applied to predict the exit probabilities in another roundabout. As a prerequisite task, we aim to compute a measure of similarity of the exit probability models for different roundabouts. In turn, we aim both at (i) forming clusters of 'similar' roundabouts, where the potential for knowledge transfer is increased, and (ii) extracting potential generic patterns in the geometric or traffic features of roundabouts which can be associated with a higher 'similarity' of their exit probability models. As such, a first requirement for the similarity metric is that it should give a general measure of the dependence between two exit probability models, i.e., the theoretical amount of knowledge which could be transferred from one roundabout to another. What is more, directly comparing accuracy metrics is not satisfactory as it is biased by the chosen validation set, and two exit probability models with equivalent accuracy rates may nonetheless produce differing predictions.

Mutual Information

Mutual Information (MI) is defined for a pair of random variables and measures the mutual dependence between them. It quantifies the amount of information that can be obtained about one random variable from the observation of the other. MI was initially introduced by C. Shannon in [195].

MI is symmetric and non-negative, with $I(X,Y) = 0$ iif X and Y are independent random variables. The MI is defined as:

$$I(X,Y) = H(X) - H(X|Y)$$

$I(X,Y)$ is the MI of X and Y, $H(X)$ the *entropy* of X and $H(X|Y)$ the entropy of X conditioned on Y. The entropy $H(X)$ of a random variable X can be defined as the average level of information which is inherent to the possible values taken by X. If X is a discrete random variable which can take values among $(x_i)_{i\in[1,n]}$ with probabilities $(P(x_i))_{i\in[1,n]}$:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \cdot log_2(P(x_i))$$

The unit of $H(X)$ with base 2 logarithm is the shannon bit. Considering an additional random variable Y taking values in $(y_j)_{j\in[1,m]}$, the conditional entropy $H(X|Y)$ is defined as:

$$H(X|Y) = -\sum_{i=i}^{n}\sum_{j=1}^{m} P(x_i,y_j) log_2(\frac{P(x_i,y_j)}{P(y_j)})$$

Furthermore, MI can be computed between different classifiers performing the same task, i.e., ML models which give class labels to entries of the same input features. MI has been used as a metric of diversity between classifiers in applications of classifiers combination in [197] and [198]. The MI between two classifiers c_1 and c_2, classifiers of the same input features, can be computed through the following process:

1. A set of observations $K \equiv (k_i)_{i\in[1,l]}, l \in \mathbb{N}$ is provided. Each observation k_i can be passed as input to c_1 and c_2 to produce a class label, respectively, $c_1(k_i)$ and $c_2(k_i)$.

2. The set of predictions of c_1 and c_2 on K observations, i.e., $C_1 \equiv (c_1(k_i))_{i\in[1,l]}$ and $C_2 \equiv (c_2(k_i))_{i\in[1,l]}$, can be assimilated to two random variables named C_1 and C_2.

3. In turn, the mutual information of the two classifiers is expressed as $I(C_1, C_2)$.

MI quantifies the amount of information which can theoretically be obtained about the outputs of one classifier from another. As such, it can be used to compute clusters of similar roundabouts, and identify potential patterns in roundabouts which produce highly dependent classifiers, as a first step before performing knowledge transfer between the roundabouts. MI matches the requirements for a metric of similarity between two exit probability models. In turn, we compute the pairwise MI between the sets of probability estimations of the models trained on the roundabouts listed in Table 6.1.

For each pair of roundabouts (R_1, R_2) extracted from the roundabouts list in Table 6.1, with $R_1 \neq R_2$:

1. An exit probability model M_{R_1} is trained from 5000 entries extracted from the track data of R_1. A distinct evaluation set E_{R_1} of 1000 entries is also extracted from the track data. The same process is performed for R_2 to obtain a M_{R_2} trained exit probability model and a E_{R_2} evaluation set for R_2.

2. A shared evaluation set of 2000 entries is obtained by gathering the evaluation sets extracted from R_1 and R_2, $E_{R_1,R_2} = E_{R_1} \cup E_{R_2}$. The evaluation set E_{R_1,R_2} is then randomly shuffled.

3. For each evaluation entry $(e_i)_{i\in[1,2000]}$ of E_{R_1,R_2}, exit probability predictions are performed by respectively applying the M_{R_1} and the M_{R_2} trained exit probability estimation models. The predictions of M_{R_1} for each e_i are stored in the $(m1_i)_{i\in[1,2000]}$ set of values, and the predictions of M_{R_2} in $(m2_i)_{i\in[1,2000]}$.

4. $(m1_i)$ and $(m2_i)$ can be assimilated to the outcomes of random variables $M1$ and $M2$ associated with the predictions of M_{R_1} and M_{R_2}. In turn, their MI can be computed, which we note $I(M1, M2)$.

5. To perform the actual computation of the MI between the probability predictions of exit probability estimators trained on the two roundabouts $I(M1, M2)$, we use the EDGE estimator as introduced in [199].

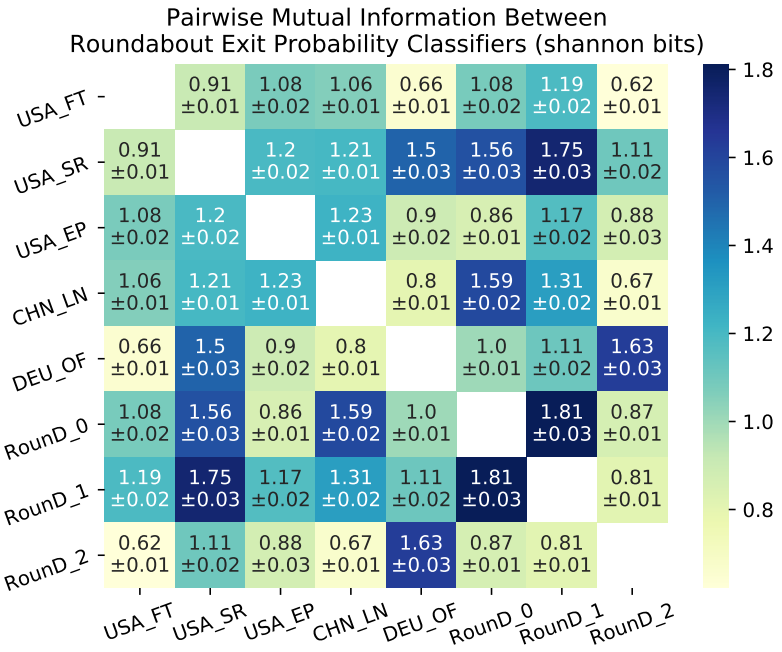


Figure 6.6: Pairwise Mutual Information of Exit Probability Classifiers

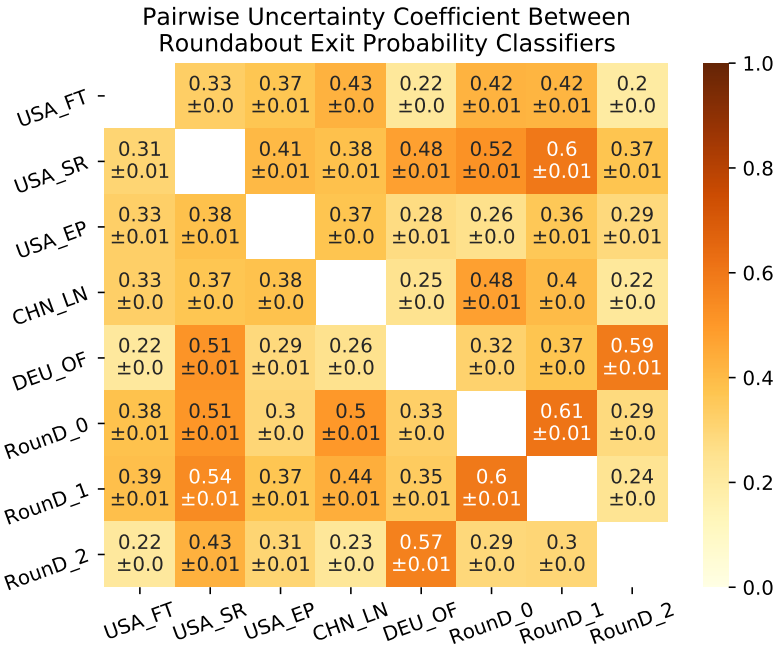


Figure 6.7: Pairwise Uncertainty Coefficient of Exit Probability Classifiers

Figure 6.6 illustrates the obtained values of pairwise MI for the exit probability classifiers associated with the considered roundabouts. For clarity reasons, the identifiers of INTERACTION roundabouts have been shortened through the removal of the *'DR_'* and *'Roundabout_'* characters. The results are presented in a heatmap matrix. Each value of pairwise MI has been computed 20 times with different sampling of training and evaluation entries for the exit probability models in order to compute 95% confidence intervals. The obtained values provide hints on the level of similarity between the exit probability models of the different roundabouts. Yet, it is challenging to interpret and compare the similarity differences using the MI values as-is. As such, we proceed to a normalization of MI values to ease their interpretation.

Normalization

The MI values as presented in Figure 6.6 are positive but not bounded. As such, it is challenging to interpret the significance of the differences of the obtained MI values. A normalization procedure is required which produces bounded values, e.g., in the $[0, 1]$ interval, where 0 represents independent random variables, and 1 represents dependent random variables such that the knowledge of one variable completely determines the other.

The Uncertainty Coefficient (UC) is a normalized MI metric which matches the aforementioned requirements. The UC of the random variable X given Y is defined as:

$$UC(X|Y) = \frac{I(X,Y)}{H(X)}$$

It comes from this definition that the UC is a normalized variant of the MI in the $[0, 1]$ interval, as:

1. $UC(X|Y) \geq 0$, as $I(X,Y) \geq 0$ and $H(X) \geq 0$.
2. $UC(X|Y) \leq 1$, as $I(X,Y) = H(X) - H(H|Y) \implies I(X,Y) \leq H(X)$.

The $UC(X|Y)$ value can be interpreted as the fraction of shannon bits which can be predicted from X provided full knowledge of the outcomes of Y. It provides a scaled level of similarity between random variables. Unlike MI, the UC is not symmetric as the amount of information which can be obtained about X given Y may differ from the amount of information which can be obtained about Y given X.

Values of pairwise MI $I(M1, M2)$ for pairs of distinct roundabouts (R_1, R_2) extracted from Table 6.1 were computed and shown in Figure 6.6. To obtain the $UC(M1|M2)$ and $UC(M2|M1)$ values, a prerequisite is to compute the entropy values $H(M1)$ and $H(M2)$.

As introduced in Paragraph 6.1.2 the Shannon entropy is defined for random variables taking a discrete set of possible values as outcome. Yet the probability

estimation model produces continuous probability values by default. To simplify the computation, the probability values produced by $M1$ and $M2$ are discretized by rounding the obtained values to one digit. The set of possible outcomes of $M1$ and $M2$ is then reduced to 11 values in $(\frac{p}{10}, p \in \{0, 1, ..., 10\})$. In turn, the entropy of $M1$ can be computed as $H(M1) = -\sum_{p=0}^{10} P(M1 = \frac{p}{10}) \cdot log_2(P(M1 = \frac{p}{10}))$, and similarly for $H(M2)$. This discretization is also used to compute MI values.

Figure 6.7 illustrates the obtained values of pairwise UC for the exit probability classifiers associated with the considered roundabouts. Unlike MI, UC values are not symmetric: the UC value $UC(M1|M2)$, with $M1$ and $M2$ the random variables respectively associated with the predictions of classifiers trained on the R_1 and R_2 roundabouts can be found in the (R_1, R_2) cell, with R_1 the roundabout identifier on the y-axis and R_2 the identifier on the x-axis. For example, the UC obtained on RounD_0 given USA_FT is 0.38 ± 0.01, whereas the UC obtained on USA_FT given RounD_0 is 0.42 ± 0.01. Like with the MI computation, 95% confidence intervals are provided which were computed from 20 computations for each pair of roundabouts with different random samplings of training and evaluation data.

In addition to the comparison of the levels of MI information between roundabout exit probability classifiers, the UC provides an understanding of the amount of information which can be obtained from one classifier knowing the predictions of another. The highest values of UC can be obtained for the pairs of roundabouts (RounD_0, RounD_1) and (DEU_OF, RounD_2), with about 60% of similarity. On the other hand, `USA_FT` and `RounD_2` only feature about 20% of similarity to each other.

6.1.3 Roundabout Semantic Description and Mutual Information

From the computation of the pairwise MI and UC of the exit probability classifiers of the roundabouts listed in Table 6.1, we obtain a 'distance matrix' as illustrated by Figure 6.6. We use the MI for this distance matrix as, unlike the UC, it is symmetrical. In turn, this distance matrix can be used to produce clusters of similar roundabouts, using algorithms such as hierarchical clustering [200] or DBSCAN [201], which require a distance matrix as input.

Figure 6.8 illustrates a possible clustering of the roundabouts based on the MI similarity of their associated exit probability classifiers. To begin with, a new distance matrix is computed by inverting the MI values of Figure 6.6. Each value MI is replaced by $\frac{1}{MI}$. As such, lower values indicate higher similarity between the associated roundabout classifiers. Then, a hierarchical clustering is performed using the *ward* linkage method, as described in [200]. The results are presented under the form of a dendogram graph, which presents clusters as well as their level of similarity. For example, it shows that RounD_0 and RounD_1 are highly similar, with a MI value of $\frac{1}{0.55} \approx 1.81$, as shown in Figure 6.6. As such, they have been grouped in a cluster. Depending on the desired number of clusters, other similar roundabouts can also be included, such as USA_SR or CHN_LN. On the other hand, RounD_1 has less similarity with the DEU_OF and RounD_2 roundabouts.

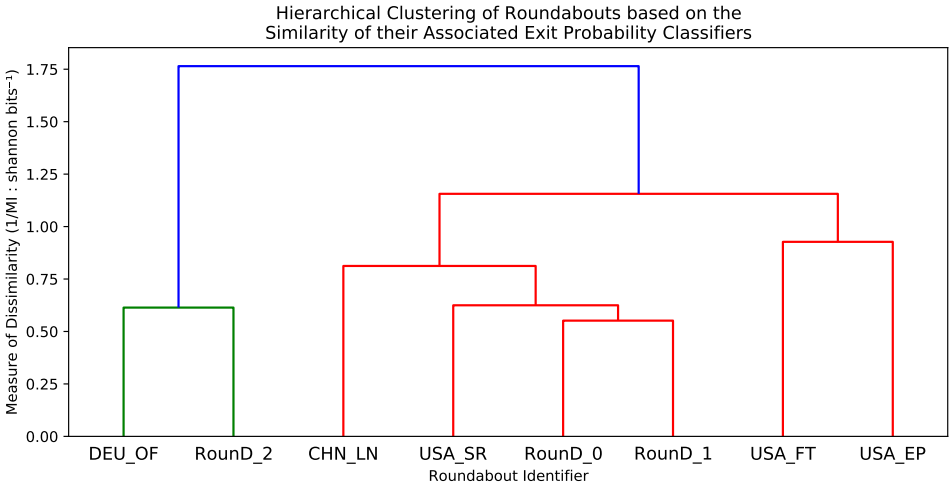


Figure 6.8: Clustering of Roundabout Exit Classifiers by MI Similarity

The computation of clusters of similar roundabouts allows the creation of groups of roundabouts among which knowledge transfer for exit probability models is likely to be more accurate, i.e., potentially allowing the training of a model based on data obtained from another similar roundabout or the application of a model on another similar roundabout. Yet, the computation of similarity clusters from known values of MI supposes that a model has already been trained for each roundabout to produce predictions and in turn compute MI values. In turn, this limits the interest of knowledge transfer, as knowledge is already available for each considered roundabout.

As such, in this section, we aim to go one step further and explore the reasons which make two roundabouts feature high MI/UC values. Namely, we select a set of geometry and traffic-based semantics to describe each roundabout, and aim to identify semantic characteristics linked to increased UC values between two roundabouts.

Geometric Semantics

To begin with, we investigate if links can be inferred between the geometric characteristics of two roundabouts and their UC values. In order to relate geometric differences to UC values, a table is produced which for each pair of roundabouts (R_1, R_2) associates a set of geometric differences of R_1 and R_2 with $UC(R_1|R_2)$. The considered geometric differences are computed from the values listed in Table 6.1. Namely, we consider the quantities which we believe may have the strongest impact on the exit behavior of vehicles in a roundabout. This includes human-related factors which we represent by the country of the roundabout, and topological factors. Namely, we consider the following quantities:

1. Whether the two roundabouts are located in the same country.
2. $\Delta Entries$: The absolute difference of the number of entry legs of both roundabouts.

Table 6.2: Training Set for the Relationship between Roundabout Geometry Differences and UC

Pair of Roundabouts	**Same Country**	**ΔEntries**	**ΔLanes**	**ΔRadius**	**ΔWidth**	***UC***
USA_FT / USA_SR	True	3	0	4.50	4.50	*0.31*
USA_SR / USA_FT	True	3	0	4.50	4.50	*0.33*
USA_FT / USA_EP	True	3	0	2.25	2.25	*0.33*
USA_EP / USA_FT	True	3	0	2.25	2.25	*0.37*
USA_FT / CHN_LN	False	3	1	14.00	0.00	*0.33*
CHN_LN / USA_FT	False	3	1	14.00	0.00	*0.43*
USA_FT / DEU_OF	False	4	0	0.25	4.50	*0.22*
		...				
RounD_0 / RounD_1	True	0	1	7.25	4.50	*0.60*
RounD_1 / RounD_0	True	0	1	7.25	4.50	*0.61*
RounD_0 / RounD_2	True	1	1	8.50	4.50	*0.29*
RounD_2 / RounD_0	True	1	1	8.50	4.50	*0.29*
RounD_1 / RounD_2	True	1	0	1.25	0.00	*0.30*
RounD_2 / RounD_1	True	1	0	1.25	0.00	*0.24*

3. $\Delta Lanes$: The absolute difference of the number of driving lanes in the circular part of both roundabouts.

4. $\Delta Radius$: The absolute difference of the radius of both roundabouts in meters. The radius of a roundabout is defined as the distance between its center point and the beginning of the asphalt of its circular part.

5. $\Delta Width$: The absolute difference of the width of both roundabouts in meters. The radius of a roundabout is defined as the width of the asphalt in its circular part.

Table 6.2 illustrates an extract of the 56-entry produced table associating geometric differences of pairs of roundabouts with their resulting UC value. In turn, we analyze potential relationships between the geometric features listed in Table 6.2 and their associated UC values through two complementary approaches. On the one hand, a correlation analysis is performed between geometric features and UC values. On the other hand, a decision tree is trained to associate the features of geometric differences with UC values. It is a supervised ML approach using Table 6.2 as a training set. Due to its explainability, the trained decision tree model can be used to infer patterns and links between geometric features and UC values.

Correlation Analysis On the one hand, we perform a correlation analysis between each feature of geometric difference for pairs of roundabouts and their associated UC value. Figure 6.9 illustrates the correlation matrix obtained on the training set shown in Table 6.2. We notice a moderate negative correlation between $\Delta Entries$ and UC values, with a correlation coefficient of -0.46. This indicates that a lower difference in entry numbers between roundabouts tends to be associated with higher UCs between the corresponding exit probability classifiers. Figure 6.10 details this negative correlation through scatter plots of pairs of $\Delta Entries$ and UC values, and includes descriptions of their distribution. It shows that roundabouts which have

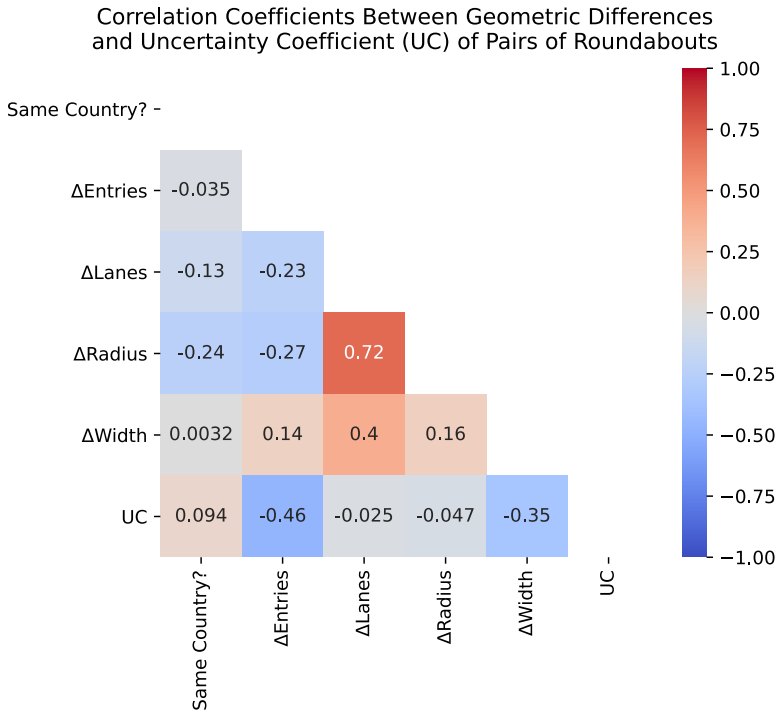


Figure 6.9: Correlation between Roundabout Geometric Differences and UC

the same number of entries feature higher UC values in average. On the other hand, roundabouts with a different number of entries feature lower average UC values. In turn, the difference in the number of entries can be identified as an important factor influencing the UC similarity of the exit probability classifiers of two distinct roundabouts. In parallel, the difference of the width $\Delta Width$ of two distinct roundabouts also features a low negative correlation with the associated UC values, with a correlation coefficient of -0.35. Other geometric differences are uncorrelated with UC values of pairs of roundabouts. To summarize, low differences in the number of entry legs and the width of the circular parts of two roundabouts can be associated with a higher probability of high UC values.

Regression Tree Analysis On the other hand, we train a regression tree to estimate UC values based on the geometric features listed in Table 6.2. Figure 6.11 shows the obtained regression tree. It provides an explainable set of rules to estimate UC values based on conditions of geometric semantic values. In turn, the observation of the rules produced by the trained regression tree provides additional hints on the relationship between geometric features and UC values. Similarly to the results obtained through correlation analysis, $\Delta Entries$ as well as $\Delta Width$ can be identified as important features to relate to UC values. What is more, $\Delta Radius$ should also be considered, provided preconditions on $\Delta Entries$. Namely, if two roundabouts feature the same number of entries, a radius difference $\Delta Radius \leq 8.12m$ can be associated with higher UC values, i.e., 0.5 in average. On the other hand, greater radius differences produce lower UC values, with an average of 0.36. If two roundabouts do not feature the same number of entries, $\Delta Width$ is the decisive factor as pairs of roundabouts featuring $\Delta Width \leq 1.125m$ feature a UC of 0.38 in average, against 0.29 for other pairs.

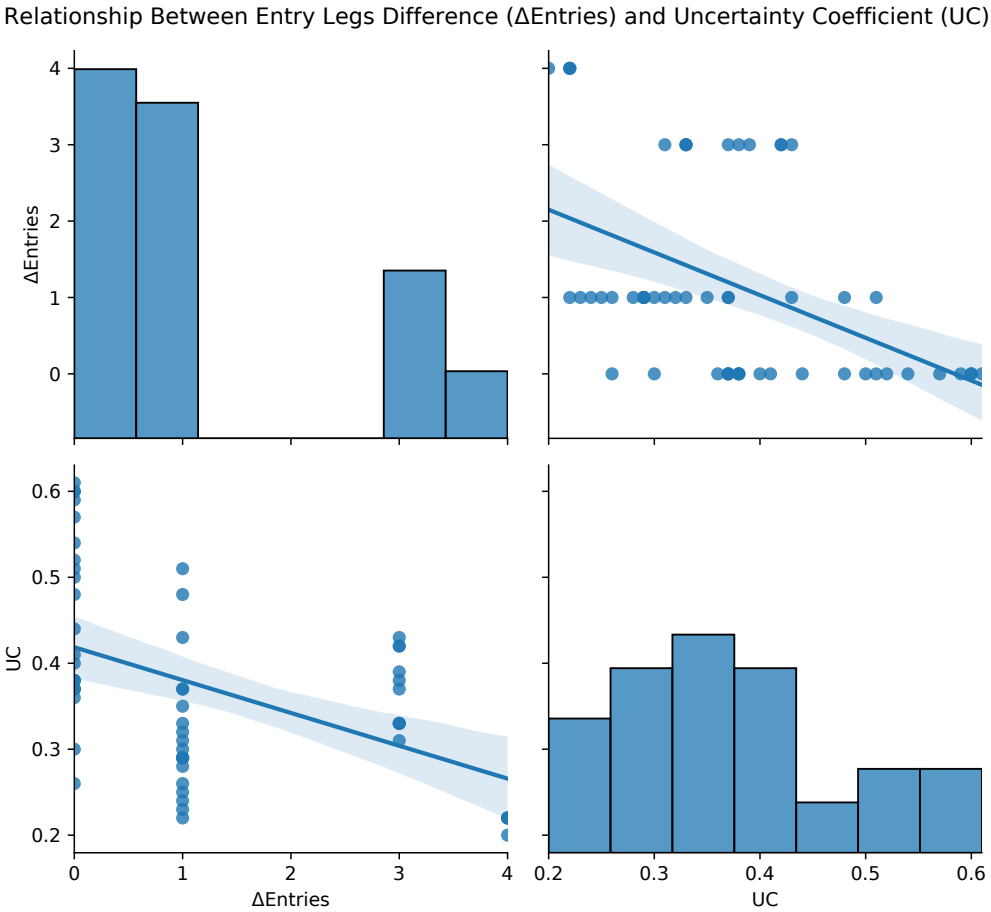


Figure 6.10: Relationship between Roundabout Entry Legs Difference and UC

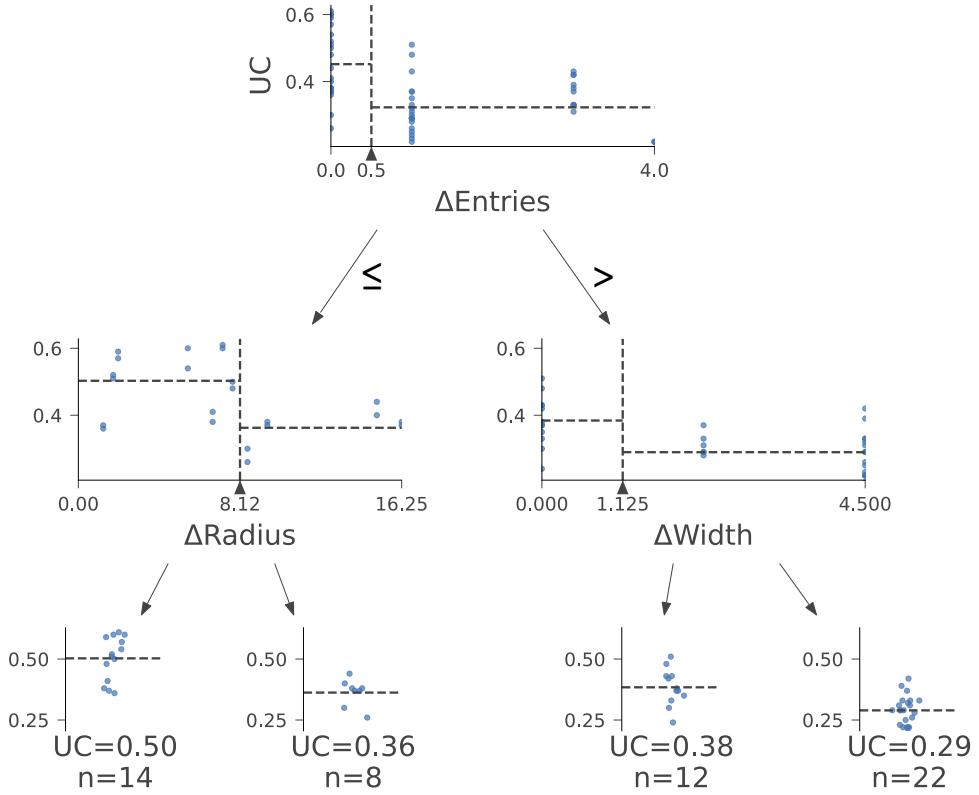


Figure 6.11: Regression Tree to Estimate UC values from Roundabout Geometric Differences

As such, $\Delta Entries$, $\Delta Width$, $\Delta Radius$ can be defined as important factors having an impact on the UC value obtained for the exit probability classifiers associated with a pair of roundabouts. The highest UC values are obtained for pairs of roundabouts featuring the same number of entry legs and an absolute radius difference of less than $8.12m$.

Traffic Semantics

In this section, we study whether similar traffic conditions in two distinct roundabouts can lead to a higher similarity, i.e., UC values for the pair of roundabouts. Namely, we aim to estimate if traffic conditions are an important explanatory factor of variations of UC values for pairs of roundabouts. A prerequisite to estimate the impact of traffic conditions on UC values is to compute normalized values representing traffic conditions in two distinct roundabouts, which can be accurately compared. In turn, provided with a normalized metric to estimate and compare traffic conditions in roundabouts, we partition the track data in each roundabout by the traffic conditions in which it was collected, and train the associated exit probability classifiers. UC values are computed for pairs of roundabout exit classifiers trained on each roundabout identifier and traffic conditions. Finally, a correlation analysis is performed to exhibit a potential correlation between the differences in traffic conditions in two roundabouts and their associated UC value.

To begin with, we compute a normalized metric to represent the traffic condi-

tions as the level of congestion in a roundabout. The computation of such a metric can in turn be used to determine if exit probability classifiers trained on two distinct roundabouts with similar traffic conditions feature higher UC values than with different traffic conditions. A desired property of the selected metric is that it should be adapted to each roundabout. Namely, a similar value of traffic flow may lead to congestion in a small-scale roundabout, yet not in a larger roundabout. As such, the values of flow in a roundabout should be normalized by a measure of its capacity, to allow comparison with roundabouts of other geometric shapes.

To address the aforementioned requirements, we select the *volume to capacity ratio* or *v/c ratio* metric as a normalized indicator of the level of congestion. It is a standard metric used in previous works, such as [202], which estimates its impact on freeway safety. It is defined as the flow of vehicles divided by the capacity of an intersection. As such, to compute the *v/c ratio* traffic congestion level metric, we compute both (i) the flow at each entry leg and (ii) the *capacity*, of each considered roundabout. In turn, we normalize the sum of the entry flows at each entry leg with the capacity of the roundabout.

On the one hand, the *flow* of an entry leg can be defined as the number of vehicles entering the circular part of the roundabout through that leg per time unit, e.g., in vehicles per hour. On the other hand, *capacity* can be defined as the maximum flow, i.e., sum of the flow of all entry legs, which can be attained in a roundabout before saturation in a given traffic context.

Flow Computation To measure the flow of vehicles entering a roundabout through a given entry leg, we define 'sensing areas' in each roundabout. Figure 6.12 illustrates the sensing areas associated with each entry of the `CHN_LN` roundabout in green-colored boxes. For each recording of vehicle tracks associated with a roundabout, the track data is replayed and divided into sections $(T_i)_{i>0}$ of a given time length t_{sub}. In turn, for each time subdivision (T_i), the number of unique vehicles which entered the roundabout through any entry, i.e., green-colored box in Figure 6.12, is counted and divided by t_{sub}. The obtained value is the entry flow of the roundabout and is equal to the sum of the flow at each of its entry legs.

Capacity Computation Several models have been described to estimate the capacity of a roundabout based on the observed traffic conditions, as summarized in [203, 204]. As the considered roundabouts listed in Table 6.1 are largely located in the USA or Germany, we consider the two following approaches:

- A roundabout capacity computation approach focusing on USA-based roundabouts has been defined in The Highway Capacity Manual (HCM), published by the Transportation Research Board in the USA. It was lastly updated in 2016, as described in [205]. We refer to this approach as 'HCM2016'. Equation 6.1 is given to compute the capacity of an entry lane of a roundabout using the HCM2016 approach.

$$C = A \cdot exp(-B \cdot Q_c) \tag{6.1}$$

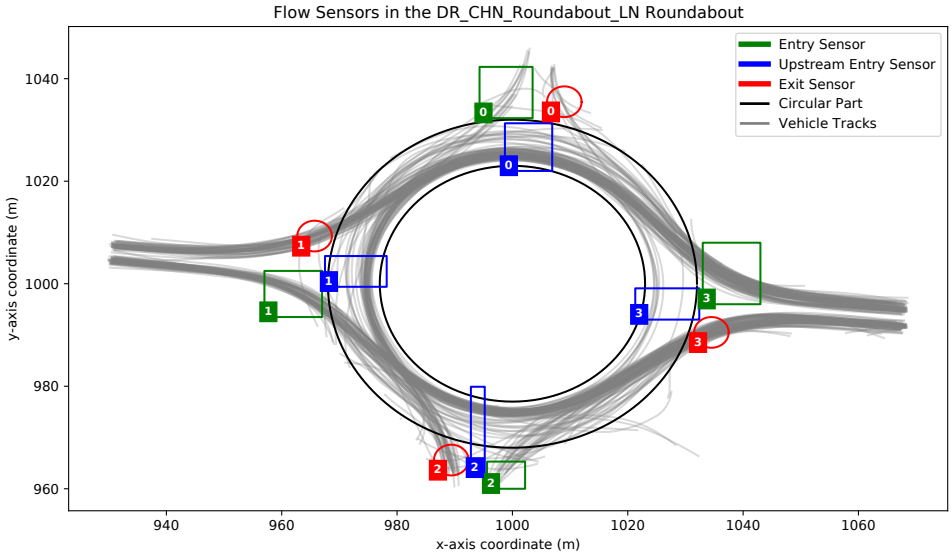


Figure 6.12: Example of Sensing Areas Defined to Compute Flow Values in a Roundabout

Where:

- C is the capacity of the considered entry in vehicles per hour.
- $A = \frac{3600}{t_f}$, $B = \frac{t_c - \frac{t_f}{2}}{3600}$
- t_f represents the follow-up headway and t_c the critical headway in the roundabout. Values of t_f and t_c are provided in the HCM for different contexts, i.e., the number of lanes in the entry and the number of circular lanes in the considered roundabout.
- Q_c is the flow in *vehicles per hour* upstream the entry, i.e., excluding the vehicles which exited the roundabout in the same leg. Q_C is measured by counting the number of vehicles which cross the associated blue-colored box as shown in Figure 6.12, in a given period of time.

- Alternatively, another model was defined considering German roundabouts as described in [206], which we refer to as the 'German' approach. Equation 6.2 is given to compute the capacity C of an entry lane of a roundabout using the German approach.

$$c = \left(1 - \frac{\Delta \cdot q_c}{n_c}\right)^{n_c} \cdot \frac{n_e}{t_f} \cdot exp(-(t_0 - \Delta) \cdot q_c) \tag{6.2}$$

Where:

- $c = \frac{C}{3600}$ is the capacity of the considered entry in *vehicles per second*.
- n_e and n_c are the number of lanes, respectively, in the considered entry leg and in the circular part of the roundabout.

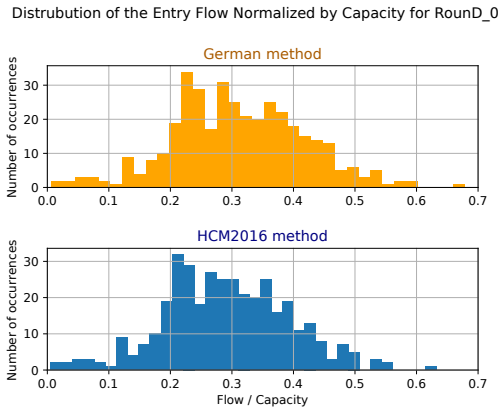


Figure 6.13: Distribution of Traffic Congestion Metric Values in RounD_0

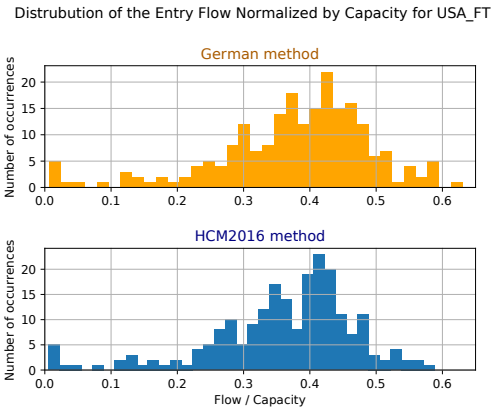


Figure 6.14: Distribution of Traffic Congestion Metric Values in USA_FT

- $t_0 = t_c - \frac{t_f}{2}$, Δ is the minimum time headway in the circular lanes of the roundabout. Values are provided, namely $t_c = 4.12s$, $t_f = 2.88s$ and $\Delta = 2.10s$.
- $q_c = \frac{Q_c}{3600}$ is the flow in *vehicles per second* upstream the entry, i.e., excluding the vehicles which exited the roundabout in the same leg.

For each of the considered roundabouts, we split each available recording into sequential 1-minute long subdivisions. For each subdivision, we compute the flow as well as the capacity of the roundabout, respectively, as the sum of the flow and the capacity of each entry of the considered roundabout. The capacity of the roundabout is computed both using the HCM2016 and the German method. In turn, we normalize the roundabout entry flow value by its capacity to obtain a normalized traffic congestion metric for each 1-minute subdivision, i.e., the *v/c ratio*, which we note γ. Figures 6.13 and 6.14 illustrates the distribution of the γ normalized traffic congestion metric values computed for each available minute of recording, respectively, for the `RounD_0` and `USA_FT` roundabouts. The distribution is shown both using the German capacity computation in orange and the HCM2016 capacity computation in dark blue. We notice that the German and the HCM2016 capacity computation approaches produce similar distributions of γ values and do not yield significantly different results, as illustrated by Figures 6.13 and 6.14. As such, we select one of the approaches to compute traffic congestion metrics for all roundabouts, e.g., the German approach.

UC Analysis Each roundabout is associated with a list of 1-minute long subdivisions for which a value of traffic congestion has been computed. We note $R \equiv (r_i)_{i \in \{0,\dots,\|R\|\}}$ the set of subrecording data for a roundabout. For each subrecording r_i, the associated traffic congestion metric is noted γ_i. In turn, 1-minute subdivisions are grouped with other subdivisions featuring a similar value of traffic congestion metric. Namely, for each roundabout, a partition of R is defined as $R = [R_{0.0 \leq \gamma < 0.1}, R_{0.1 \leq \gamma < 0.2}, R_{0.2 \leq \gamma < 0.3}, R_{0.3 \leq \gamma < 0.4}, R_{0.4 \leq \gamma < 0.5}, R_{0.5 \leq \gamma < 0.6}]$, with $\forall x, y \in [0,1], x < y, \forall i \in \{0, \dots \|R\|\}, x \leq \gamma_i < y \iff r_i \in R_{x \leq \gamma < y}$. As the number of subrecordings which feature $\gamma > 0.6$ is negligible in the available roundabouts, the partition does not consider values of γ greater than 0.6.

Table 6.3: Training Set for the Relationship between Roundabout Traffic Differences and UC

Pair of Exit Probability Models	**Same Roundabout**	$\Delta\gamma$	***UC***
(RounD_2, $0.2 \leq \gamma < 0.3$) / (RounD_1, $0.3 \leq \gamma < 0.4$)	False	0.1	0.28 ± 0.003
(USA_SR, $0. \leq \gamma < 0.4$) / (USA_SR, $0.2 \leq \gamma < 0.3$)	True	0.1	0.59 ± 0.006
(USA_FT, $0. \leq \gamma < 0.4$) / (USA_EP, $0.1 \leq \gamma < 0.2$)	False	0.2	0.35 ± 0.001
(RounD_1, $0.2 \leq \gamma < 0.3$) / (RounD_2, $0.2 \leq \gamma < 0.3$)	False	0.0	0.26 ± 0.002
(RounD_0, $0.1 \leq \gamma < 0.2$) / (USA_FT, $0.2 \leq \gamma < 0.3$)	False	0.1	0.40 ± 0.001
(RounD_0, $0.5 \leq \gamma < 0.6$) / (RounD_1, $0.3 \leq \gamma < 0.4$)	False	0.2	0.53 ± 0.002
(USA_EP, $0.2 \leq \gamma < 0.3$) / (DEU_OF, $0.4 \leq \gamma < 0.5$)	False	0.2	0.31 ± 0.003
...			
(USA_EP, $0.2 \leq \gamma < 0.3$) / (CHN_LN, $0.1 \leq \gamma < 0.2$)	False	0.1	0.41 ± 0.003
(USA_FT, $0.3 \leq \gamma < 0.4$) / (DEU_OF, $0.3 \leq \gamma < 0.4$)	False	0.0	0.23 ± 0.001
(CHN_LN, $0.1 \leq \gamma < 0.2$) / (RounD_0, $0.1 \leq \gamma < 0.2$)	False	0.0	0.48 ± 0.003
(CHN_LN, $0.0 \leq \gamma < 0.1$) / (USA_EP, $0.2 \leq \gamma < 0.3$)	False	0.2	0.41 ± 0.003
(USA_FT, $0.2 \leq \gamma < 0.3$) / (USA_EP, $0.1 \leq \gamma < 0.2$)	False	0.1	0.35 ± 0.002
(RounD_0, $0.3 \leq \gamma < 0.4$) / (USA_FT, $0.4 \leq \gamma < 0.5$)	False	0.1	0.40 ± 0.002

Thus, for each roundabout, vehicle track data is separated and available for various discrete levels of traffic congestion. In turn, an exit probability model is trained from the track data associated with each level of congestion. It is no longer one exit probability model that is associated with each roundabout, e.g., RounD_0, but 6 different models trained from track data sensed in increasing levels of traffic congestion, e.g., (RounD_0, $0 \leq \gamma < 0.1$), ..., (RounD_0, $0.5 \leq \gamma < 0.6$). What is more, we set a minimum number of training entries to train a model, i.e., if less than 5000 entries are available as training data for a given (Roundabout, γ) pair, no model is trained for this pair.

Then, as in Section 6.1.2, the pairwise UC of each trained exit probability model is computed, along with a 95% confidence interval obtained from 20 different samplings of training and validation data. Then, we perform a correlation analysis of the difference of traffic conditions in pairs of roundabouts with the associated UC value. Namely, a table is produced which associates different traffic conditions in roundabouts with UC values. Table 6.3 shows an extract of the produced table of 552 rows. The 'Same roundabout' column indicates whether the two considered models have been trained from track data extracted from a single roundabout, i.e., where only the traffic conditions differ. The $\Delta\gamma$ column refers to the absolute difference between the congestion levels of two roundabouts, i.e., given a pair of models $[(R_1, a \leq \gamma < b), (R_2, c \leq \gamma < d)]$, $\Delta\gamma = |a - c| = |b - d|$.

Finally, as in Section 6.1.3, a correlation analysis is performed between the features of Table 6.3, i.e., 'Same Roundabout', $\Delta\gamma$, and the associated UC values. No correlation is found between the traffic conditions in two situations and the UC of the associated exit probability models. Namely, the correlation coefficient between $\Delta\gamma$ and UC is -0.035 and 0.068 respectively when the rows featuring the 'Same Roundabout' are excluded and included. As such, similar traffic conditions between two roundabouts are not related to higher UC values, and neither are different traffic conditions to lower UC values. On the contrary, 'Same Roundabout' and 'UC'

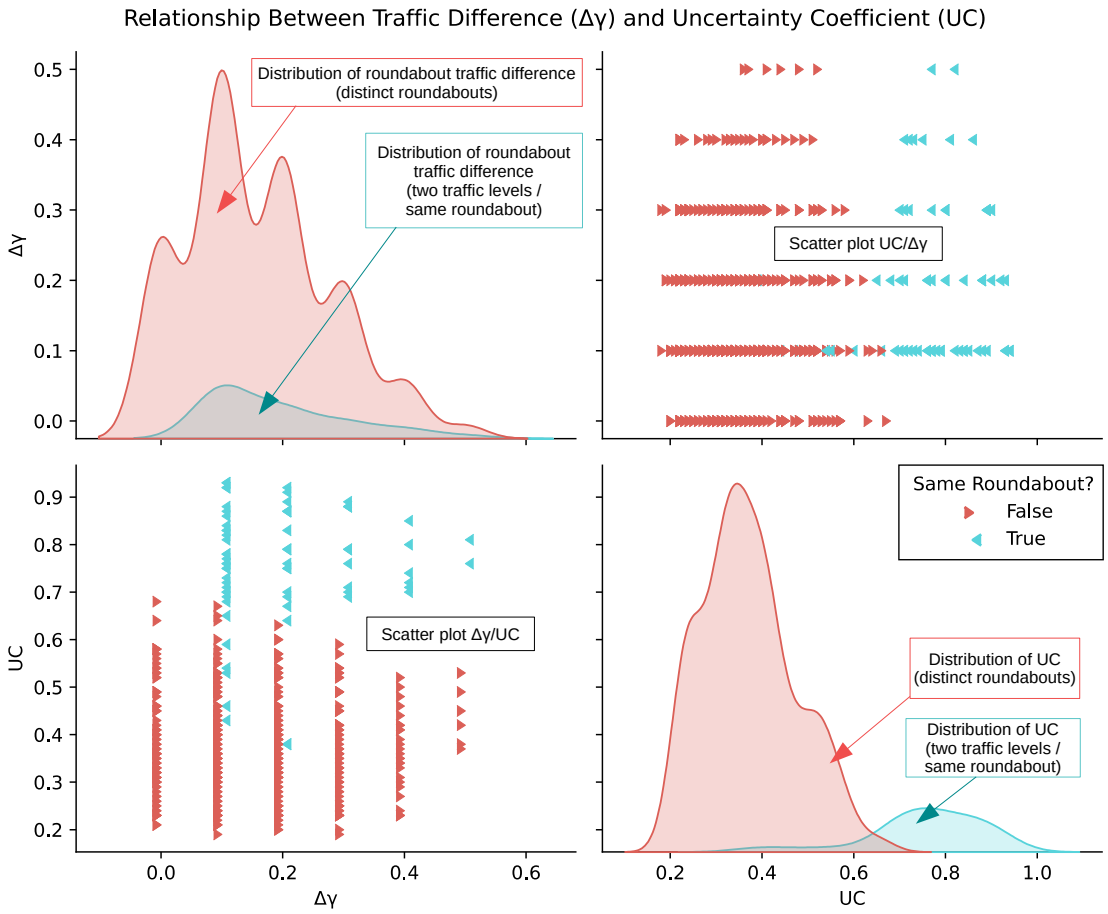


Figure 6.15: Relationship between Roundabout Traffic Difference and UC

feature a correlation coefficient of 0.77, seemingly indicating that inherent geometric features of a roundabout impact UC values more than traffic differences.

Figure 6.15 details the relationship between $\Delta\gamma$ and UC values. It contains two scatter plots at the bottom-left and top-right corners, which plot the pairs of $(\Delta\gamma, UC)$ values. The top-left and bottom-right graphs, respectively, illustrate the distribution of $\Delta\gamma$ and UC values. Moreover, the data is split into two groups: (i) Exit probability models which were computed from a single roundabout with different traffic conditions are shown in light blue, and (ii) other models, which were computed from different roundabouts with potentially different traffic conditions, are shown in red. Figure 6.15 illustrates the lack of correlation between the differences in traffic conditions and the UC values of the associated exit probability models.

The inherent geometric features of roundabouts seem to impact UC values more than traffic conditions, as two models trained on different traffic conditions of any scale in a single roundabout largely feature higher UC values than models trained on two different roundabouts, regardless of the similarity in traffic. Yet, it can be observed that the distinct roundabouts featuring the highest values of UC also feature similar traffic conditions, as the maximum values of UC obtained in the set are decreasing when $\Delta\gamma$ increases. As such, while traffic conditions do not give indications on UC values, they seem to be a prerequisites to attain high UC values, i.e., here, in the $[0.65, 0.7]$ range.

Discussion

In this section, we investigated the factors which can be used to pertinently describe the context of training and use of exit probability models, based on the roundabout in which they were trained. Namely, the identified context description factors can be used to assess the information theoretic similarity between two contexts, e.g., as the difference in radius or width between two roundabouts lowers, the UC values between their associated exit probability models increase.

As a next step, in Section 6.1.4, we contribute the definition of similarity conditions to assess if the contexts of two roundabouts are similar. In turn, we evaluate the impact of (i) using an exit probability model in a context which is similar to its context of training, and (ii) training an exit probability model by completing the training data from roundabouts featuring a similar context.

6.1.4 Impact of Semantic-Based Knowledge Distribution

In the last section, we identified semantic rules to assess the similarity of two roundabouts in terms of their associated exit probability model. Namely, an equal number of entries as well as a radius difference of less than $8.12m$ can be associated with higher UC values. On the other hand, traffic congestion level were not found to have an impact on the similarity of two roundabouts.

In this section, we investigate whether the semantic conditions, which were identified as a factor of similarity between the exit probability models of two roundabouts,

can in practice improve the accuracy of model application and training in unseen roundabouts. For example, when crossing an unknown roundabout, autonomous vehicles may wish to obtain the probability of exit of vehicles on that roundabout to facilitate its crossing. Yet, no known model has been trained for this specific unseen roundabout. As such, we evaluate whether the semantic description of the roundabout, i.e., its number of entries, width, and radius, can be used to select similar existing exit probability models trained on other roundabouts which predict exit probabilities accurately on the unseen roundabout.

What is more, we investigate to what extent the semantic criterion of similarity between roundabouts can be used to improve the accuracy of model training on an unseen roundabout. Namely, when crossing an unseen roundabout where no or few training data is available, to what extent can the available training data be completed using training data gathered on distinct, semantically similar roundabouts? In turn, what is the accuracy of the obtained model compared to completing the training data with randomly selected training data from other roundabouts?

Model Application

In this subsection, we investigate whether the knowledge of the geometric semantics of roundabouts can improve the accuracy of exit probability estimation in an unknown roundabout. Namely, when encountering an unknown roundabout, we assess whether exit probability models trained on semantically similar roundabouts can predict vehicle exits with higher accuracy than models extracted from random roundabouts.

To begin with, we define a set of three alternative semantic conditions to determine if two roundabouts are considered *similar* or not. As defined in Figure 6.11, the most similar pairs of roundabouts are characterized by (i) the same number of entry legs, and (ii) low differences in radius. What is more, lower difference in width between two roundabouts is correlated with higher similarity values. Based on these findings, we define three alternative binary conditions to flag whether two roundabouts should be considered similar or not. The three conditions are defined in Equations 6.3, 6.4, and, 6.5, ordered from the stricter to the most inclusive condition.

$$\Delta Entries = 0 \wedge \Delta Radius \leq 2.0m \wedge \Delta Width \leq 2.0m \tag{6.3}$$

$$\Delta Entries = 0 \wedge \Delta Radius \leq 6.0m \tag{6.4}$$

$$\Delta Entries = 0 \wedge \Delta Radius \leq 8.12m \tag{6.5}$$

For each of the similarity conditions defined in Equations 6.3, 6.4, and, 6.5, we assess the applicability of exit probability estimation models trained on each roundabout to their associated similar roundabouts. Namely, for each similarity equation E and for each roundabout R_{target} listed in Table 6.1:

1. For each roundabout R in Table 6.1 such that $R \neq R_{target}$, the accuracy of the exit probability model trained on R and applied on R_{target} is assessed as follows:
   (a) A set S_{target} of 1000 validation entries is extracted from the track data of R_{target}.
   (b) The exit probability model trained for R is used to predict whether the vehicle will exit at the next exit or not, for all the validation entries listed in S_{target}, i.e., $p_{exit} > 0.5 \iff$ *the vehicle will exit at the next exit.*
   (c) The predictions of R are matched with the ground truth, i.e., whether the observed vehicles actually exited the roundabout at the next exit. Various accuracy metrics are computed. Namely, the accuracy, precision and F1-score.
2. The results are grouped in two sets, i.e., grouping roundabouts which are *similar* and *distant* to R_{target}. Namely, iif $E(R, R_{target})$ then R is flagged as *similar* to R_{target}, and iif $\neg E(R, R_{target})$ then R is flagged as *distant* to R_{target}.

The metrics of accuracy which we use to assess the applicability of the exit probability model trained on R to track data in R_{target} is detailed as follows:

- *Accuracy* is defined as the number of correct predictions over the size of the validation set, i.e., $\|S_{target}\| = 1000$.
- *Precision* is defined as $\frac{tp}{tp+fp}$, with fp the number of false positives, i.e., cases when an exit was predicted but did not happen, and tp the number of true positives, i.e., correct predictions of roundabout exits. As such, precision is a metric which penalizes false positives. It is relevant in this exit prediction case study as, in an autonomous driving case, a vehicle predicting the exit behavior of other vehicles to choose whether to enter a roundabout will merely lose time when wrongly assessing that a vehicle will stay in the roundabout. On the other hand, a collision risk may occur in case of false positives where a vehicle is expected to exit but does not in practice.
- The *F1-score* is defined as the harmonic mean between precision and recall, which is equivalent to the precision applied to true and false negatives. As such, recall penalizes false predictions of vehicles staying in the roundabout. It does not lead to danger but can lead to slow downs if vehicles repetitively assess that vehicles will stay in the roundabout and stand by before crossing a roundabout. As such, the F1-score features a balance between precision and recall.

Figure 6.16 illustrates the *accuracy* metrics obtained when applying *similar* and *distant* exit probability models in a target roundabout. The similarity condition defined in Equation 6.3 is used in Figure 6.16, i.e., the most strict condition. Each column, i.e., x-label, of the Figure is associated with a target roundabout. For each target roundabout, the obtained accuracy metrics obtained when applying *distant* roundabouts in red and *similar* roundabouts in blue are presented vertically. For reference, the accuracy of the exit probability model trained on the target roundabout is

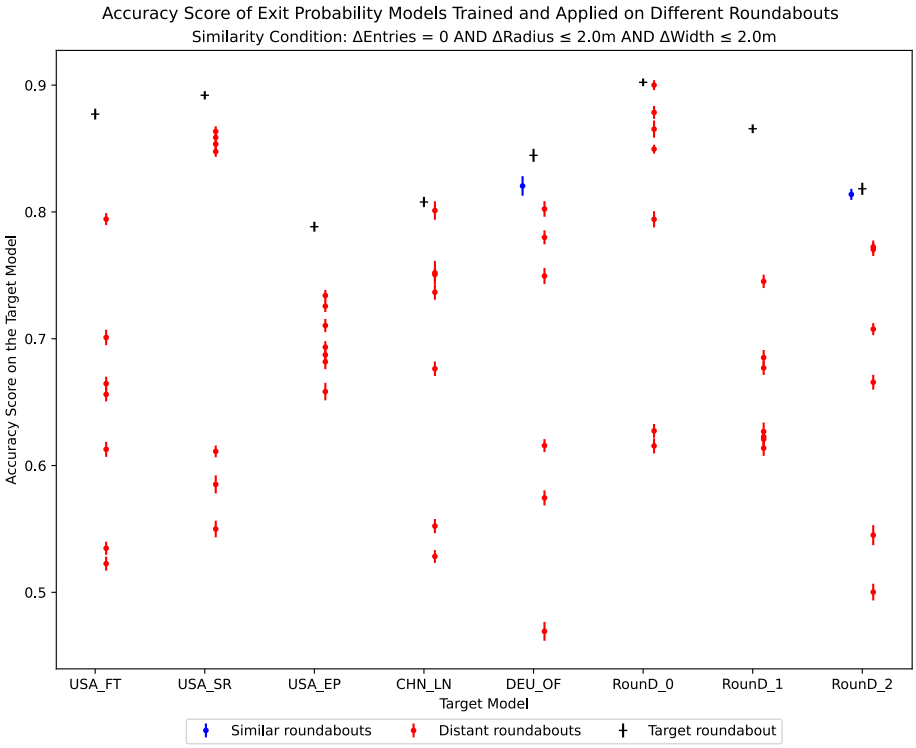


Figure 6.16: Accuracy of Exit Probability Models Applied to Distinct Roundabouts using Similarity Equation 6.3

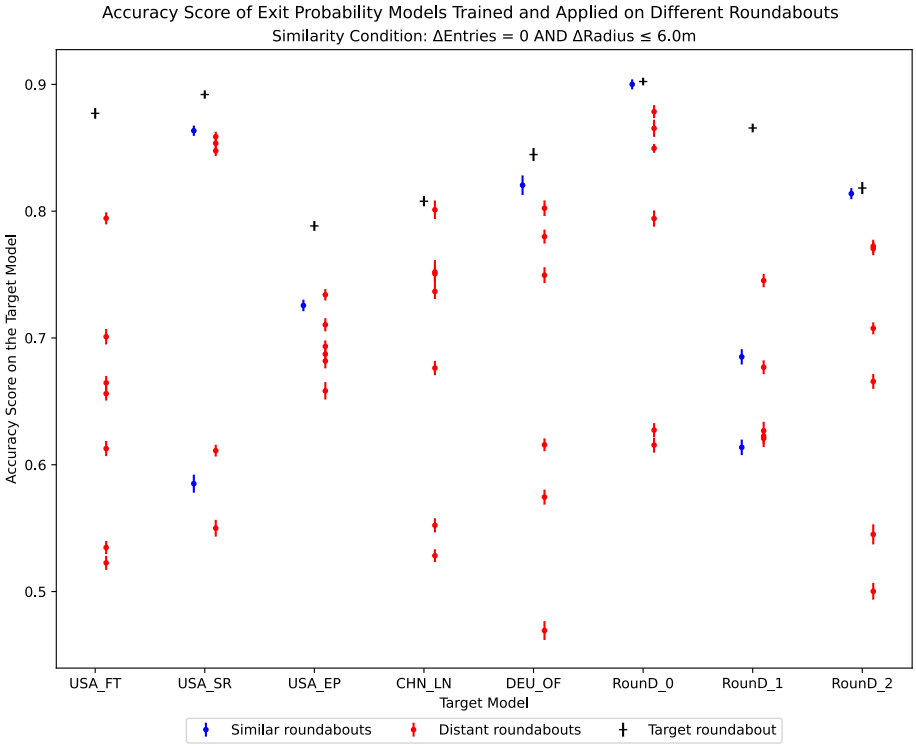


Figure 6.17: Accuracy of Exit Probability Models Applied to Distinct Roundabouts using Similarity Equation 6.4

provided in black. Each accuracy point is associated with a 95% confidence interval computed after averaging 20 computations using different sampling of training and validation data. As illustrated by Figure 6.16, the only pair of roundabouts which matches the similarity criteria as defined by the strict Equation 6.3 is (DEU_OF, RounD_2). In turn, models matching the considered strict similarity condition can be associated with a higher accuracy than non-similar models when applying exit probability models to a target roundabout. Similar benefits are observed for the precision and F1 scores using this similarity condition.

While Figure 6.16 illustrates higher accuracy scores for a pair of similar roundabouts, the similarity condition defined in Equation 6.3 is strict, limiting the number of similar roundabouts to the (DEU_OF, RounD_2) pair. In turn, Figure 6.17 illustrates the accuracy scores obtained using Equation 6.4 as a similarity condition. The considered Equation 6.4 similarity condition is less conservative than Equation 6.3, as it removes the width constraint, which has less impact on similarity than the radius difference according to Figure 6.9. What is more, it allows greater radius differences to be considered similar. We notice that, using this less conservative similarity condition, situations occur where exit probability models trained on similar roundabouts cannot be associated with higher accuracy, precision, and F1 scores, i.e., in the `USA_EP` and `RounD_1` cases.

While less strict similarity conditions allow more roundabouts to be flagged as similar, it also introduces noise in the results, where some exit probability models do not produce higher accuracy scores than *distant* models on similar roundabouts. As such, selecting a random similar exit probability model to be applied on an unknown roundabout cannot be guaranteed to yield accurate results, using a moderately strict similarity condition. As such, we consider an approach where the predictions from several similar exit probability models are gathered to produce an exit probability value, to mitigate the case where a single moderately similar exit probability model is selected, which does not yield accurate results.

Thus, in addition to the scores of each individual model applied to a target roundabout, we evaluate the accuracy scores of the gathered predictions of all the exit probability models which are similar to a target roundabout. Namely, the grouped prediction of the set of similar models on a target roundabout R_{target} is computed using a *voting regression* technique as follows:

- To evaluate each entry $s \in S_{target}$, the 1000-entry evaluation set associated with R_{target}:
    1. The probability p_i of exit associated with s is assessed by each of the similar exit probability models M_i.
    2. The probability of exit associated with the ensemble of exit probability models p_{voting} similar to R_{target} is computed as the average of the probabilities p_i produced by each individual similar model. This approach is an ensemble learning technique known as *voting regression*, as described in [207].
    3. The obtained p_{voting} probability is used to predict whether the vehicle will exit at the next exit, i.e., $p_{voting} > 0.5 \iff$ *the vehicle will exit at the next exit.*

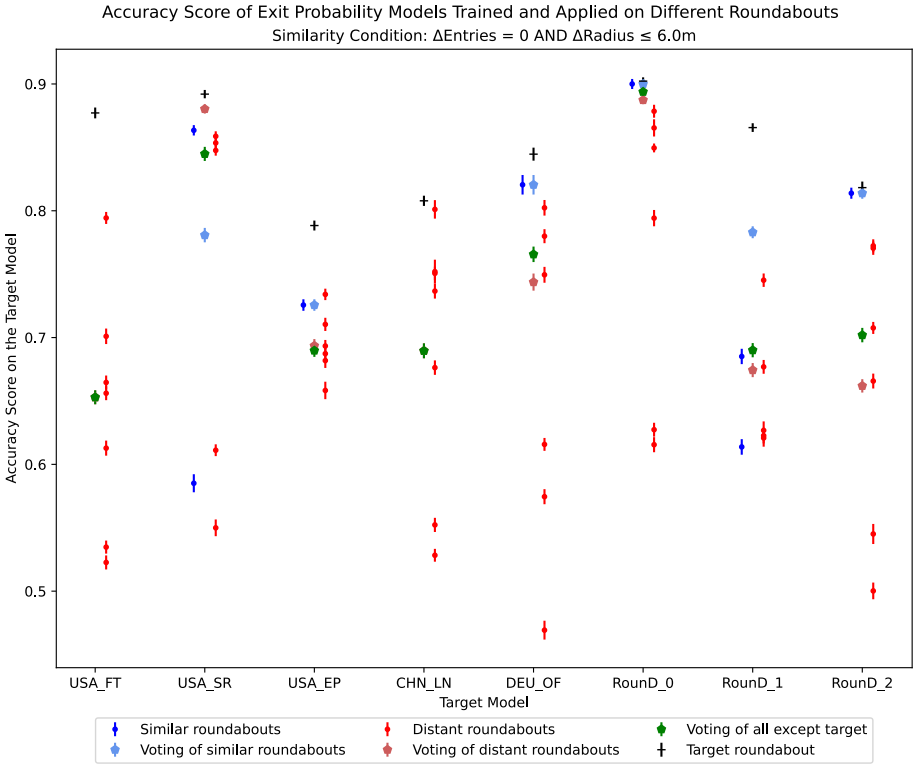


Figure 6.18: Accuracy of Ensemble Exit Probability Models Applied to Distinct Roundabouts using Similarity Equation 6.4

- In turn, (i) the ensemble voting accuracy of the gathered similar exit probability models is compared with (ii) the accuracy of the ensemble voting of the gathered *distant* exit probability models, and finally (iii) the ensemble voting of all exit probability models except the one trained on the target roundabout.

Figure 6.18 illustrates the accuracy score of the ensemble exit probability models using the similarity condition listed in Equation 6.4. By grouping the predictions of similar models through ensemble voting, higher prediction accuracy can be achieved than the prediction of the least accurate similar model. What is more, in the `RounD_1` case, ensemble voting on the two similar models allows to reach higher prediction accuracy than both models taken independently, and that any other considered exit probability model. In the USA_SR case, ensemble voting allows to mitigate the low accuracy which one of the similar models is scoring. As such, ensemble voting is a promising technique to maintain a high accuracy of exit probability prediction while using a less strict similarity condition, which allows more roundabouts to feature similarities.

Finally, Figure 6.19 illustrates the obtained results when considering the similarity condition described in Equation 6.5, which further extends the radius difference required to flag two roundabouts as *similar* to $8.12m$. It matches the similarity condition identified in the regression tree of Figure 6.9. Figure 6.19 shows a reduction of the accuracy of the ensemble-similar models, as less similar roundabouts are considered similar. In turn, similar roundabouts are not significantly more accurate than other roundabouts for the `USA_SR`, `USA_EP` and `RounD_0` target models.

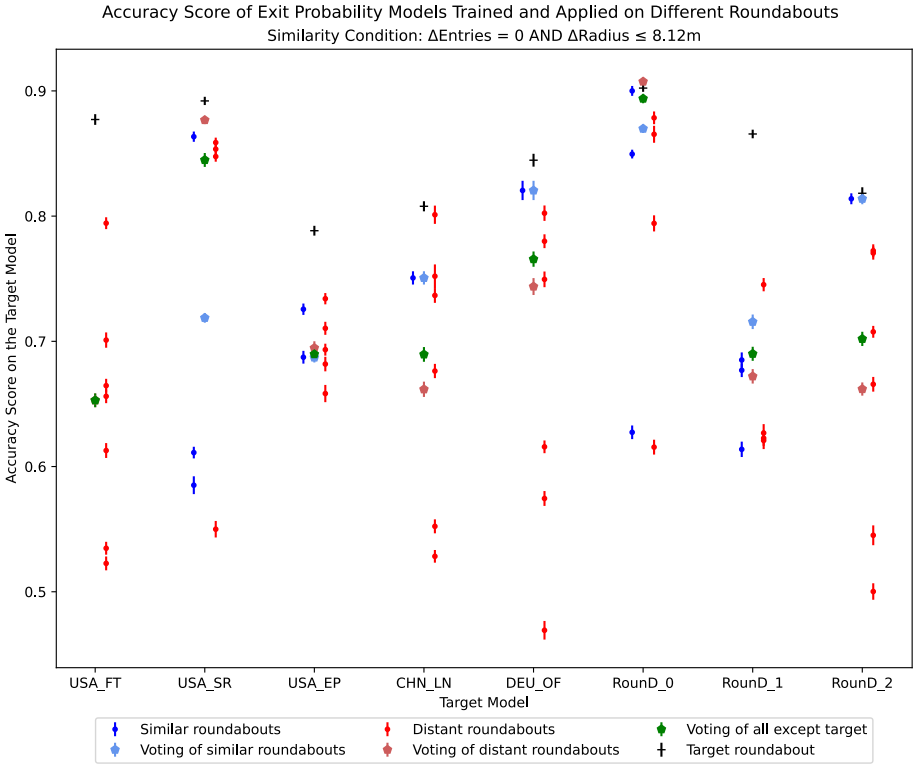


Figure 6.19: Accuracy of Ensemble Exit Probability Models Applied to Distinct Roundabouts using Similarity Equation 6.5

Thus, semantic similarities between the considered roundabouts can be associated with an increased accuracy of the exit probability model associated with one roundabout, when it is applied on the other. This increase in accuracy is especially observed using a strict similarity condition as in Equation 6.3. On the other hand, less strict similarity conditions as in Equation 6.4 lead to 'noise' as some individual exit probability models may not perform accurately on roundabouts flagged as similar, i.e., using a moderate similarity condition. Yet, this can be mitigated by gathering the predictions of exit probability models associated with similar roundabouts through ensemble voting techniques. Finally, using a weak similarity condition as defined in Equation 6.5 leads to mixed results, where similar exit probability models cannot be associated with an increased accuracy in several target roundabouts.

This illustrates the potential of semantic descriptions of models to increase the accuracy of knowledge sharing and application in vehicular networks. What is more, it demonstrates the need to fine-tune similarity conditions to each use case. Namely, a balance must be found between too strict conditions which involve too few roundabouts and too weak similarity conditions which cannot be associated with clear benefits in terms of the accuracy of similar models.

Impact & Discussion In this section, we have contributed an algorithm to select the most relevant exit probability models in a given context, as shown in a flow chart in Figure 6.20. Namely, when crossing a roundabout for which no model has been trained, the context of the roundabout is matched to the context of training of existing probability models, which are used either independently or grouped in an

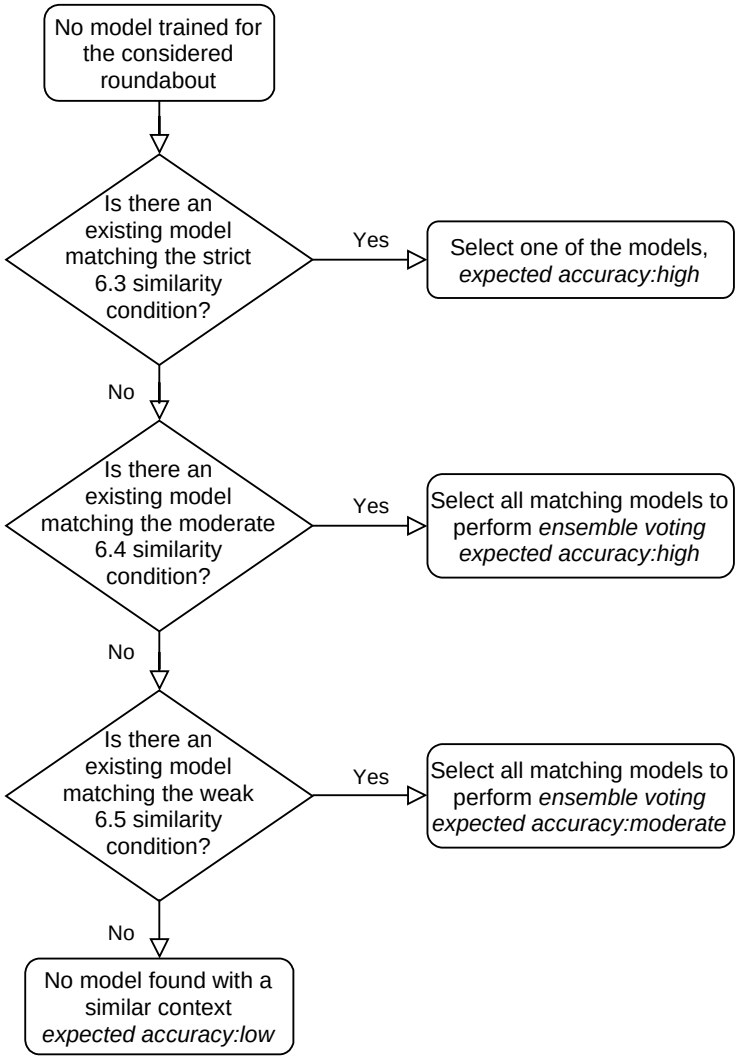


Figure 6.20: Flow Chart of the Process to Select Exit Probability Models to Use in a Specific Context

ensemble voting process, to ensure the accuracy of the created knowledge. In turn, we show that the returned models produce knowledge with high accuracy compared to the selected random models, notably using stricter similarity conditions.

In turn, this paves the way for a context-aware distribution of knowledge in vehicular networks. When a specifically targeted knowledge is not available for the exact driving context, context reasoning and semantic descriptions can be used to select distinct models which have been trained in a similar context, and as such, more accurate. Furthermore, in the next paragraphs, we investigate the impact of using the similarity of contexts to improve the accuracy of training a new model for an unknown roundabout.

Model Training

The semantic similarity between roundabouts can be used to select an accurate existing exit probability model when crossing an unknown roundabout. In turn,

we investigate on the benefits of the knowledge of the semantic similarity between roundabouts to facilitate the training of a new model tailored for an unknown roundabout.

As illustrated in Section 6.1.4, using similar models to generate exit probability knowledge for an unseen roundabout performs more accurately than using randomly selected models. Yet, the optimal accuracy is obtained for the exit probability models which were specifically trained for a target roundabout using track data extracted from it. However, training an exit probability model for a specific target roundabout requires relatively large sets of input training track data to be sensed from it. In this subsection, we investigate on the potential benefits of track data extracted from similar roundabouts to assist the training of an exit probability model for a new roundabout where limited training data is available.

As such, we contribute an algorithm to train a model for a new roundabout, partially using data extracted from other roundabout in a similar context, which is listed in Figure 6.21. In turn, we apply the algorithm for each $R_{target} \in Rs$, with Rs the set of considered roundabouts as listed in Table 6.1.

Figure 6.22 illustrates the obtained results for each roundabout which features at least one *similar* roundabout, using the similarity condition as defined in Equation 6.4. This excludes the `USA_FT` and `CHN_LN` roundabouts. Each sub-figure of Figure 6.22 shows the variation of the accuracy of exit probability models trained for each target roundabout as the proportion of training data extracted from the target roundabout grows or shrinks. The y-axis shows the accuracy scores of the trained exit probability models and the x-axis shows the proportion δ of training data extracted from the target roundabout. In each graph, the blue-colored, red-colored, and green-colored curves illustrate, respectively, the accuracy of the exit probability models trained by completing the training data with data extracted from (i) similar roundabouts only, (ii) distant roundabouts only, and (iii) all other roundabouts. Each curve is accompanied with 95% confidence intervals computed after running 20 exit probability model computations with different samplings of training and evaluation data, for each target roundabout, for each value of δ, and for the three approaches, i.e., considering similar, distant, and all other roundabouts.

The results show a significant increase of the accuracy of the trained exit probability model when the training data is completed with data from similar roundabouts, as opposed to randomly selected other roundabouts or non-similar roundabouts. On the one hand, for `USA_EP` (Figure 6.22a), `USA_SR` (Figure 6.22b), and `RounD_1` (Figure 6.22e), the increase is visible for small values of δ, until proportions of respectively, 40%, 10%, and 20% of training data extracted from the target roundabout. After this, the increasing proportion of training data from the target roundabout compensates the benefits of using similar roundabouts to complete the training data. On the other hand, for `DEU_OF` (Figure 6.22c), `RounD_0` (Figure 6.22d), and `RounD_2` (Figure 6.22f), the increase of accuracy obtained when completing the training data using similar roundabout is significant for all proportions δ of training data from the target roundabout, and allow to flatten the accuracy per δ curve, i.e., attain the optimal accuracy for low proportions of target training data δ. Namely, the optimal accuracy is obtained from $\delta = 0.3$ for `DEU_OF`, and for all values of δ for `RounD_0` and `RounD_2`. On the contrary, models trained by

1. Let Rs be the set of considered roundabouts as listed in Table 6.1, and $R_{target} \in Rs$ the roundabout for which an exit probability model is being trained.
2. We consider the case where R_{target} is a roundabout where limited amount of training data has been sensed. As such, the available training data extracted from R_{target} must be completed with existing training data extracted from other roundabouts.
3. Let $Rs_{similar}$ and $Rs_{distant}$ be, respectively, the set of roundabouts which are similar and not similar to R_{target}, excluding R_{target}, using the moderate similarity condition as defined in Section 6.1.4 and Equation 6.4.
4. Let $Rs_{others} \equiv Rs \setminus \{R_{target}\} \equiv Rs_{similar} \cup Rs_{distant}$ be the set of every other available roundabouts.
5. We aim to train an exit probability model to be applied on R_{target} using a fraction δ of training data extracted from R_{target}, and a fraction $(1 - \delta)$ of training data extracted from other roundabouts. This simulates the fact that a limited amount of training data has been sensed from R_{target}. The overall size of the gathered training data is 5000 entries, and the accuracy score is computed on 1000 distinct entries extracted from R_{target} track data.
6. Namely, for each $\delta \in \{0.0, 0.1, 0.2, ..., 1.0\}$, an exit probability model is trained using a combination of track data extracted from R_{target} and similar roundabouts in $Rs_{similar}$.
   (a) $N = 5000 \cdot \delta$ training entries are extracted from R_{target}.
   (b) $M = 5000 \cdot (1 - \delta)$ entries are extracted from roundabouts in $Rs_{similar}$. An equal amount of training data is extracted from each of the similar roundabout, i.e., $\forall R_i \in Rs_{similar}, M_i = \frac{M}{\|Rs_{similar}\|}$ entries are extracted from R_i. In turn, $M = \sum_{i=0}^{\|Rs_{similar}\|} M_i$.
   (c) The set of $N + M = 5000$ obtained entries is randomly shuffled, and used to train an exit probability model, associated with a fraction of δ training data extracted from R_{target}, completed with data from similar roundabouts. The accuracy of the obtained model is computed on a distinct evaluation set of 1000 entries extracted from R_{target}.
7. The procedure described in Item 6 is repeated, using $Rs_{distant}$ and Rs_{others} to complete $R_{transfer}$ training data.
8. As such, for each $\delta \in \{0.0, 0.1, 0.2, ..., 1.0\}$, the accuracy of exit probability models trained using a $(1 - \delta)$ fraction of training data extracted from, respectively, (i) similar, (ii) distant, and (iii) all other roundabouts is computed.

Figure 6.21: Procedure of Model Training from Training Data Partially Extracted from Distinct but Similar Contexts

completing the training data from other, randomly-selected or non-similar roundabouts score lower values of accuracy and attain optimal accuracy values for high δ proportions only, i.e., $\delta = 0.7$ for `RounD_0` and `RounD_2`, and $\delta = 1.0$ for `DEU_OF`.

Impact & Discussion In this section, we contributed a procedure to improve the accuracy of a trained model, when a limited amount of training data is available, by completing the training data with data sensed in a similar context. Figure 6.21 defines the procedure.

Furthermore, this result has a general impact in terms of knowledge networking in the vehicular environment. Based on the obtained results, context-similar training data selection mechanisms have the potential to improve the accuracy of models trained by vehicles as part of FL processes, and could extend the orchestration mechanisms which we contributed in Section 4.3. Namely, instead of requiring training nodes to possess training data sensed exactly in the right context, as considered in Section 4.3, nodes which possess training data sensed in a similar context could also be selected for training, in turn increasing the number of potential training nodes for a similar accuracy.

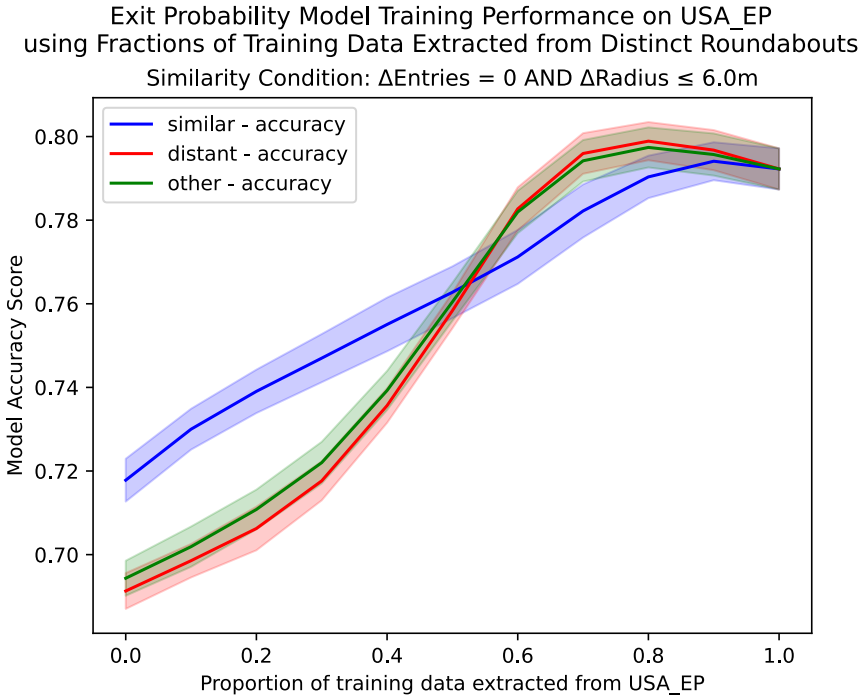
Yet, such context-aware training node selection requires the definition and networking of semantic annotations to describe not only the context of training of a model, but the conditions in which this context is similar to another, for a given model. In turn, novel applications which we could name *knowledge training as a service*, could make use of semantic annotations to both (i) understand the requirements of a generic ML training process, and (ii) match the requirements with a set of known nodes in appropriate, similar contexts, and return a list of potential training nodes.

6.2 Distribution of Roundabout Exit Knowledge: A Packet-Level Study

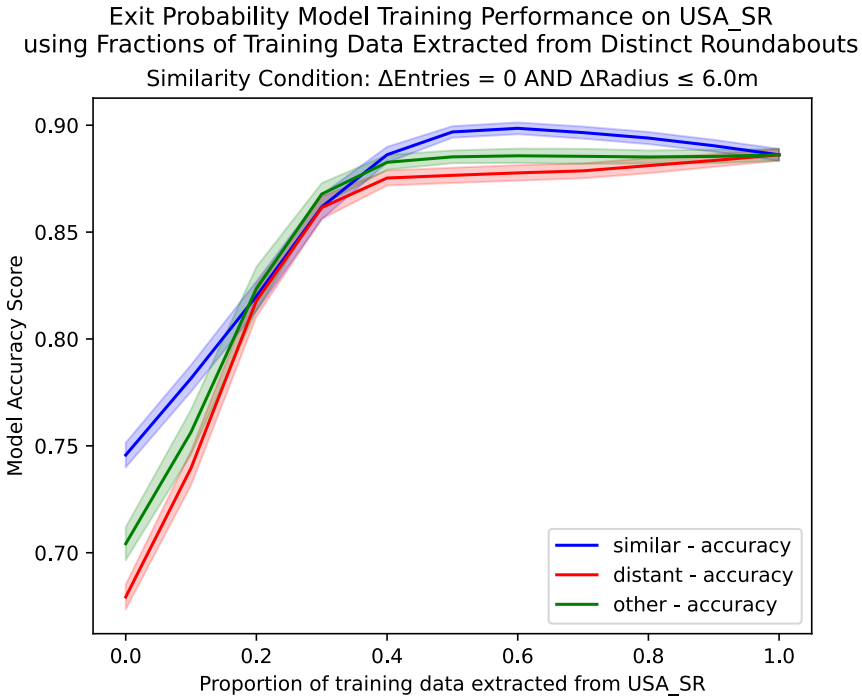
In Section 6.1, we investigated the relationship between the context in which a roundabout exit probability model is trained and its accuracy when it is used for roundabouts which feature a similar context. In this section, we use the exit probability model case to support an implementation of VKN on a packet-level simulator. The implementation replicates the vehicular mobility and involves remote exit probability knowledge creation requests based on the current context. This implementation is used to demonstrate the benefits of context-based semantic reasoning in a realistic application where vehicles and infrastructure units communicate requests for exit probability knowledge.

6.2.1 Overview

In this section, we provide an overview of the use case which we implement to evaluate context-aware knowledge distribution in vehicular networks. To begin with,

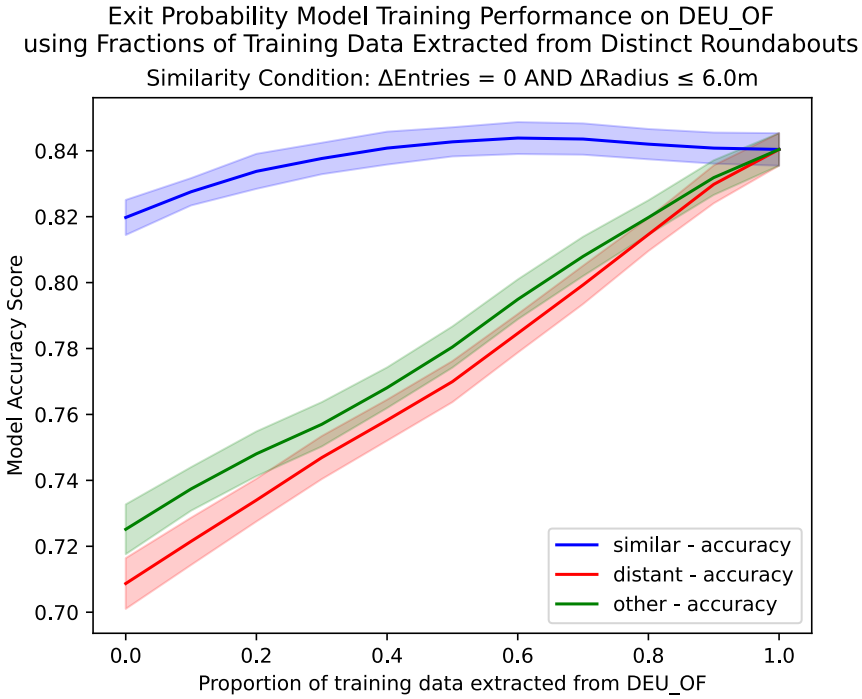


(a) DR_USA_Roundabout_EP

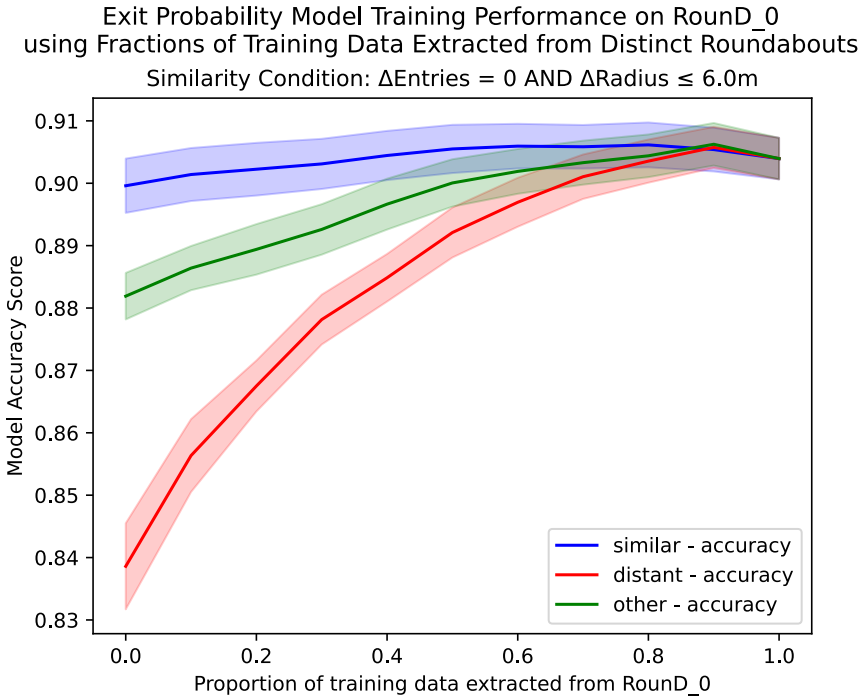


(b) DR_USA_Roundabout_SR

Figure 6.22: Accuracy of Exit Probability Models Trained using A Fraction of Track Data from Distinct Roundabouts

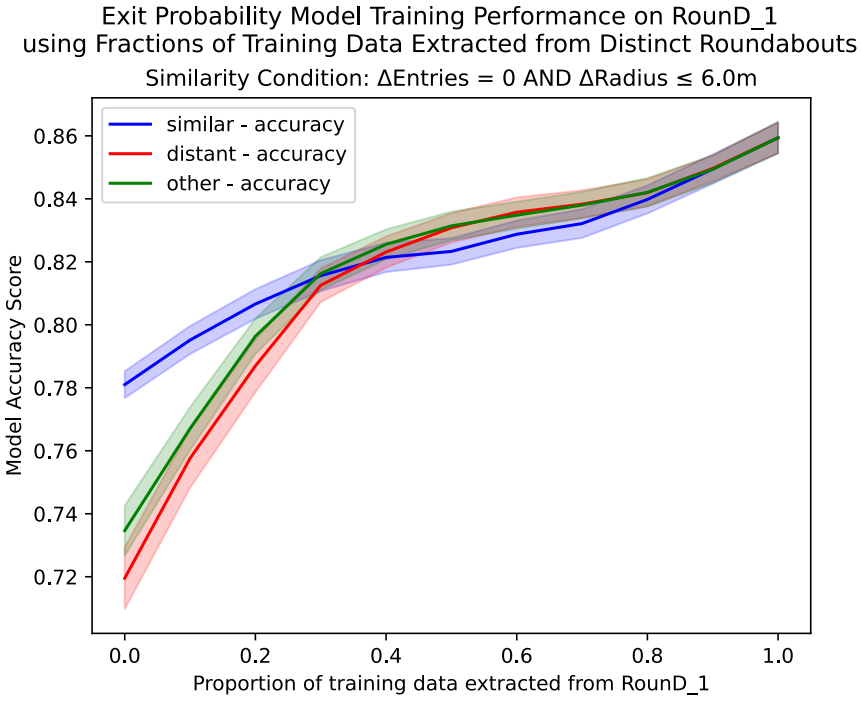


(c) DR_DEU_Roundabout_OF

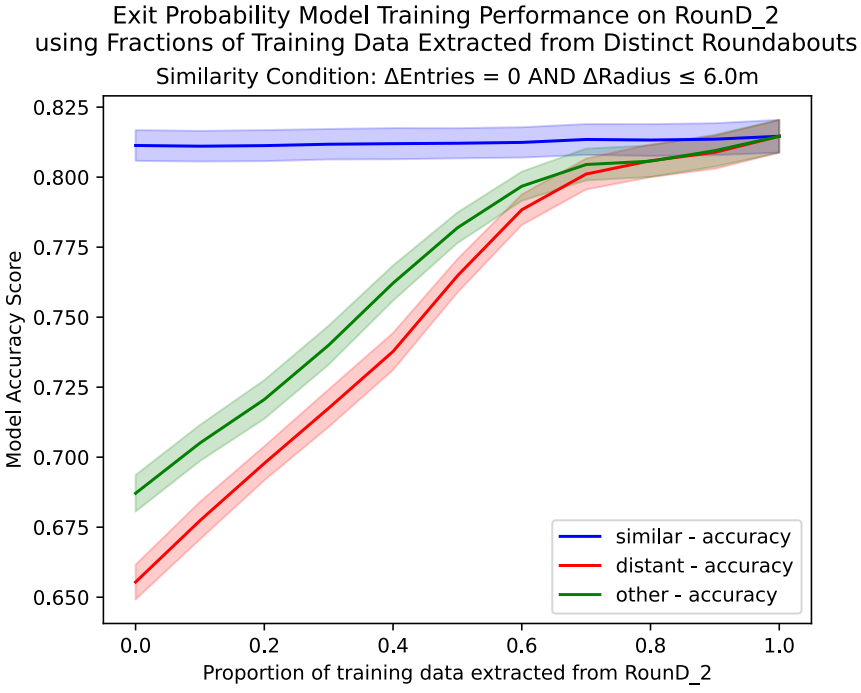


(d) RounD_0

Figure 6.22: Accuracy of Exit Probability Models Trained using A Fraction of Track Data from Distinct Roundabouts (continued)



(e) RounD_1



(f) RounD_2

Figure 6.22: Accuracy of Exit Probability Models Trained using A Fraction of Track Data from Distinct Roundabouts (continued)

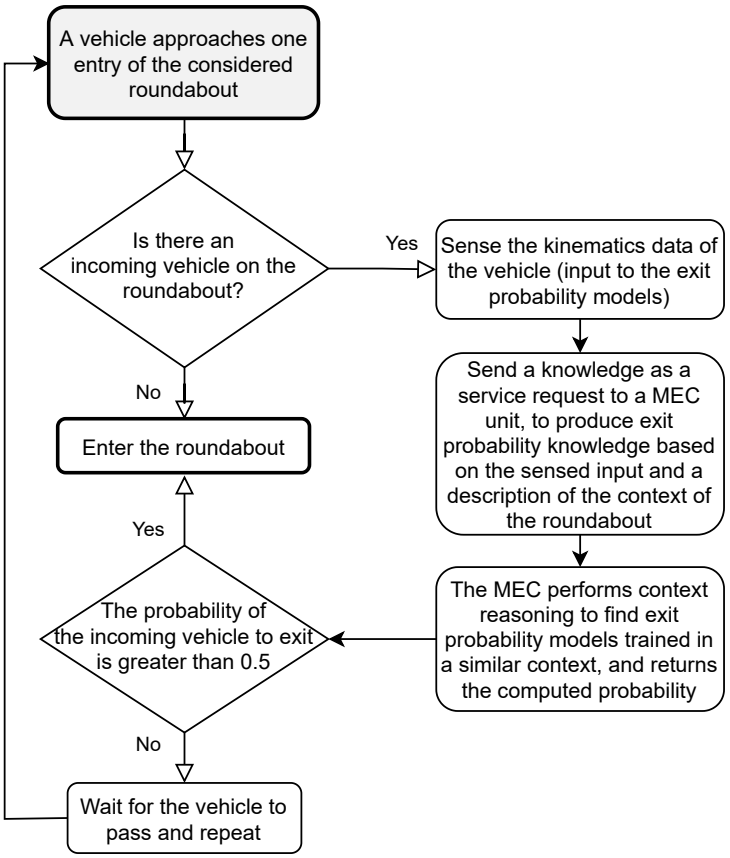


Figure 6.23: Flow chart of the Considered Vehicle Roundabout Entering Scenario

we provide a high-level description of the use case. In turn, we define a semantic description of the exit probability models defined and analyzed in Section 6.1. Then, we justify the need for a network packet-level simulation, before providing an initial overview of the simulation we implement to evaluate context-aware exit probability knowledge networking.

Use Case

We consider the following use case: As introduced earlier by Figure 6.1, vehicles which are entering a roundabout for which no exit probability model has been trained, wish to obtain the exit probability of any incoming vehicle. In turn, we contribute a procedure for vehicles to enter a roundabout safely, taking the probability of incoming vehicles to exit into account, which is represented by a flow chart in Figure 6.23. In this section, we only evaluate the knowledge networking part associated with the problem, and do not modify the mobility of vehicles to enter or not enter the roundabout. In real applications, the probability threshold to decide whether to enter the roundabout could be increased over 0.5 to increase the safety of crossing.

Thus, to obtain the exit probability knowledge about an incoming vehicle, entering vehicles formulate a *knowledge as a service* request which is disseminated to neighboring nodes. As no exit probability model is available for the roundabout being crossed, the aim of the request is to let a distant node compute the exit prob-

ability value of the sensed incoming vehicle using an existing exit probability model which has been trained in a similar context. In turn, a semantic description of the exit probability models must be used to express the knowledge retrieval requests.

Exit Probability Knowledge Semantic Description

Based on the results obtained in Section 6.1, a semantic description of the considered roundabout exit probability models can be described. Namely, the intrinsic parameters of the models are as follows:

- Inputs
  - $relative\_heading : float \in [-180, 180](deg)$
  - $relative\_distance\_to\_exit : float \in [0, 1]$
  - $relative\_lateral\_position : float \in [0, 1]$
- Output
  - $exit\_probability : float \in [0, 1]$

In turn, the extrinsic parameters, which describe the context of application of the models, are the following:

- $number\_of\_entries : integer > 0$, e.g, 4 for the `DEU_OF` model.
- $radius : float > 0(m)$, e.g, $7m$ for the `DEU_OF` model.
- $width : float > 0(m)$, e.g, $8.25m$ for the `DEU_OF` model.

A Need for Networking

We implement the aforementioned use case through a ns-3 packet simulation, to simulate the key role of networking in context-aware knowledge distribution. In turn, in Figure 6.24, we present a sequence diagram of the communication and processing performed as part of the considered exit probability knowledge distribution use case. The black-colored part represents the items which we implement as part of this study, and the gray-colored part shows additional elements to bring perspective and further illustrate the need for networking and the need for the implementation of this study on a packet-level simulation.

Namely, the entering vehicles wishing to obtain exit probability knowledge do not know exactly where any relevant exit probability knowledge models are stored. In turn, NDN is adapted in this context, as the knowledge creation requests can be disseminated without providing a destination host address. In turn, we implement a NDN-based networking of *knowledge as a service* requests. As shown in the top left corner of the figure, each knowledge creation request is conveyed in an interest packet, using a naming convention which we detail later. In turn, the request is

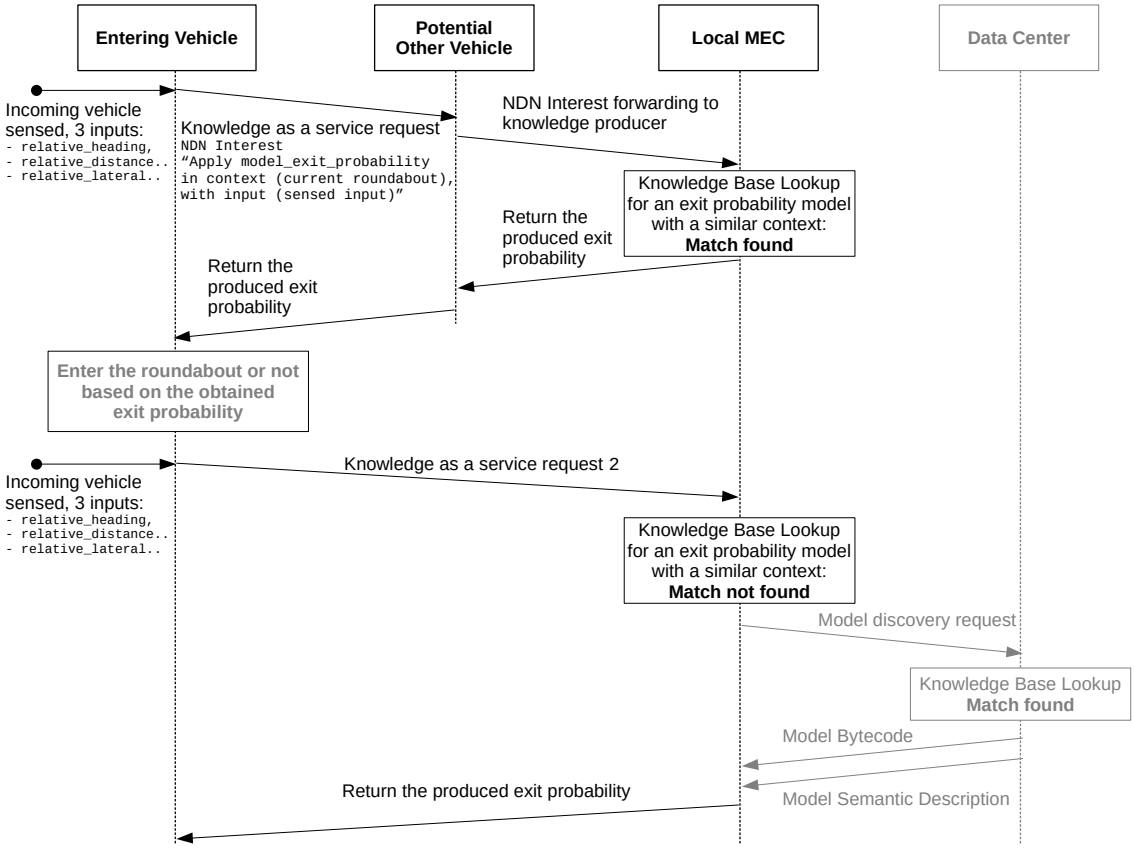


Figure 6.24: Networking Scenario for VKN Context-Aware Exit Probability Knowledge Distribution

routed to a local infrastructure node or MEC unit, which is registered as a knowledge producer for the exit probability model. The routing of *knowledge as a service* requests may be multi-hop and reach a node which is able to produce the requested knowledge after being relayed by one or several other vehicular nodes, under a fixed limit of hops.

What is more, the knowledge of exit probability models in the right context is not guaranteed to have been stored in the knowledge base of any local infrastructure node, even considering nodes reachable after several hops. Namely, local vehicular nodes may fail to possess an exit probability model trained in the right context, which would force the use of a model which was not trained in a relevant context for the knowledge creation task, thus potentially leading to reduced accuracy, as we evaluate in the next sections.

Alternatively and as a perspective, to avoid creating knowledge from exit probability models in a non optimal context, as shown in Figure 6.24, a local infrastructure node or MEC unit might send knowledge discovery messages to a remote data center which is known to hold the right knowledge, to cache the knowledge locally where it is required, at the cost of a potentially longer delay of access to the knowledge. In our publication [208], we investigate the placement of knowledge in a context where it is most needed. In turn, as future work, this could be integrated to the study, to investigate the perspectives of context-based knowledge caching.

Simulation Description

To evaluate the benefits of context-aware knowledge networking, we produce an implementation of VKN in a use case of exit probability knowledge exchange between vehicles and infrastructure nodes. The implementation is performed on the ns-3 network simulator [209]. Moreover, we use the ndnSIM 2.8 library on top of ns-3 which implements NDN networking [210].

The simulation involves two types of nodes, i.e., (i) mobile vehicles and (ii) static infrastructure nodes, e.g., MEC units. Both may communicate over a wireless channel to exchange NDN interest and content packets. As illustrated by Figure 6.1, the simulation replicates the mobility of vehicles in a roundabout, where a MEC unit has been placed in the center, and several further infrastructure nodes placed randomly around the roundabout. The vehicular mobility is extracted from the original RounD or Interaction track recordings, as listed in Table 6.1.

When an entering vehicle senses an incoming vehicle, it issues a knowledge creation request through VKN implemented over NDN, to let a remote knowledge producer produce the requested exit probability value. The remote knowledge producer is able to produce the knowledge only if it has a copy of an exit probability model in its KB. In turn, we simulate three scenarios, to evaluate the relative performance of context-aware knowledge networking with a baseline approach. As defined by Figure 6.25, we refer to the three considered scenarios as (i) context-agnostic, i.e., baseline, (ii) context-aware, without caching, and (iii) context-aware, optimized with caching.

As the scope of this application is to implement context-aware knowledge networking using the exit probability models use case, the mobility of vehicles is statically loaded in ns-3, and not modified as a reaction to the exit probabilities of incoming vehicles. As future work, mechanisms to adapt the behavior of entering vehicles based on the exit probability of incoming vehicles could be implemented. This point will be further detailed in Chapter 7. The details of the VKN implementation which allows the vehicles and the infrastructure nodes to communicate context-aware exit probability knowledge are described in the next subsection.

6.2.2 VKN Implementation

The aim of the ns-3 simulation is to let vehicles which enter a roundabout request to a remote infrastructure node, e.g., a MEC unit, the creation of exit probability knowledge about any vehicle which is already in the roundabout and may conflict with them. To implement the communications between the vehicles and the infrastructure nodes and practically allow vehicles to request the creation of exit probability knowledge to the infrastructure, we provide an implementation of VKN.

The requirements which should be addressed by the VKN implementation are both (i) to let vehicles and infrastructure units share a common understanding of exit probability knowledge creation requests, and (ii) describe semantics to convey the input kinematics data of a sensed vehicle, as well as the context linked to the current roundabout, i.e., the geometric characteristics of the roundabout identified

Context-agnostic approach (Baseline) In this approach, the remote infrastructure nodes do not take the context of usage of exit probability knowledge into account when answering knowledge creation requests from entering vehicles. Namely, exit probability models are cached in the local MEC unit in the center of the crossed roundabout. However, they are models which have not been trained in a similar context than that of the crossed roundabout. As such, a reduction of accuracy of the produced knowledge is expected.

Context-aware approach, without caching In the proposed *context-aware knowledge creation* approach (without caching), the remote infrastructure nodes take the context of usage of exit probability models into account. Namely, as no exit probability model with the right context is available in the local MEC unit, the knowledge creation requests from entering vehicles are further forwarded to remote infrastructure nodes featuring a model trained in the right context, through multicast and multi-hop wireless communications. As such, an increased accuracy of the produced knowledge is expected compared to the baseline approach.

Context-aware approach, optimized with caching In the *context-aware knowledge creation* approach without caching, requests are multi-hop forwarded to distant nodes able to produce the knowledge in the right context, which could increase the delay of access to knowledge. In turn, in a third approach, we add context-aware knowledge caching to the *context-aware knowledge creation* approach. Namely, exit probability models trained in the right context are cached in the local MEC unit, to combine the high accuracy of the produced knowledge with lower delay and overhead.

Figure 6.25: The Three Considered Knowledge Networking Scenarios

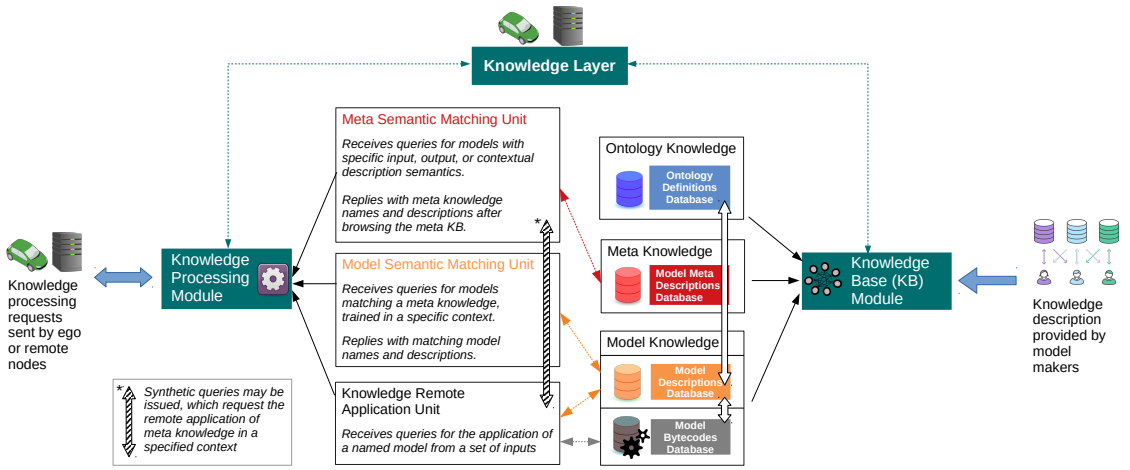


Figure 6.26: Overview of the Architecture of the VKN Implementation

in Section 6.1.

The VKN implementation which was performed involves two complementary modules, which follow the concepts introduced in Chapter 4, i.e., (i) a KB for knowledge storage, and (ii) a knowledge processing unit to receive and treat incoming knowledge creation requests, from ego or remote nodes. Figure 6.26 shows an overview of the VKN implementation. The KB and the knowledge processing modules are shown, respectively, in the right-hand and left-hand side.

Knowledge Description & KB Module

As a main contribution of this part, we contribute a novel knowledge base architecture to support the storage of multiple models which feature the same functionality and interface but have been trained in distinct contexts. Namely:

- There is a level of similarity between the `DEU_OF`, the `RounD_0`, or the `USA_EP` exit probability models, as they are all designed to produce exit probability knowledge from the same set of inputs, albeit in a different roundabout, i.e., context.
- In comparison, a model to estimate the probability of lane change in a highway would have different input parameters and context of use, and thus belong to a different class of models.

In turn, it is critical to represent these different levels of similarity between models in a KB, such that vehicles can perform a generic request, e.g., 'please produce roundabout exit probability knowledge in the context of 3-entry roundabouts, 30m radius and 10m width'. To answer such requests, the emitter must be aware of a semantic description of the *class* of models it wishes to use, and this content must be reflected in the architecture of the KB. It is not present in existing databases such as the LDM.

Namely, in Chapter 2, we presented a state of the art of the existing storage approaches in vehicular networks. In turn, in Chapter 3, we analyzed the need for

updates in these storage standards, to support the efficient storage of rich semantic context-descriptive content and knowledge. Then, in Chapter 4, we described a framework to implement the required changes and support knowledge-aware storage in vehicular networks. Finally, in this chapter, based on the results of the context analysis of roundabout exit probability models in Section 6.1, we clearly identify the required elements which should be implemented in a VKN vehicular knowledge base, as a main contribution.

Namely, we implement the following modules to store knowledge in the KB of vehicles, from the most to the least abstract, as illustrated by Figure 6.27:

- *Ontology KB:* The Ontology KB contains semantic definitions of unique names to refer to objects, as well as the type of each object and the range of values it may take.
- *Meta KB:* The Meta KB is a model description database which contains semantic descriptions which constrain a class of models featuring the same input and output items but might be trained in different contexts. The meta description has a unique name, describes the set of input and output that the models which implement the meta description will feature, as well as a list of semantic objects which are relevant to describe their context of training, and their similarity.
- *Model KB:* The Model KB contains descriptions of models which implement a meta model, trained in a specific context. Models can be described as the realization in a specific context of the knowledge interface described by a meta model.
- *Bytecode KB:* The Bytecode KB contains the machine code of trained models described in a model KB. They are used to perform the actual creation of knowledge from well-formed input.

The KB modules ensure a mutually understandable description of the interface of knowledge models as well as of their context of application. To illustrate the description of KB modules, Figure 6.28 shows the ontology, meta model, and model descriptions which we define as part of the exit probability knowledge use case, which we implement in the ns-3 simulation. The `rd_exit_proba_estimator` meta description is provided in the red-colored box on the left side of the figure. It describes the set of inputs and outputs of exit probability models, as introduced in Section 6.1 and listed in the blue-colored ontology definition box on the right side. Finally, the exit probability models associated with each of the eight roundabouts listed in Table 6.1 implement the `rd_exit_proba_estimator` meta model in the context of their associated roundabout, as illustrated by orange-colored boxes on the right side of Figure 6.28. The models are named by the lower-case name of the associated roundabout followed by the 'exit_model' suffix, e.g., `deu_of_exit_model`, or `round_0_exit_model`.

This KB architecture can in turn be used as a support for knowledge creation requests issued from entering vehicles to infrastructure units in the vicinity of the simulated roundabout. Specifically, a KB module is integrated to vehicles, which

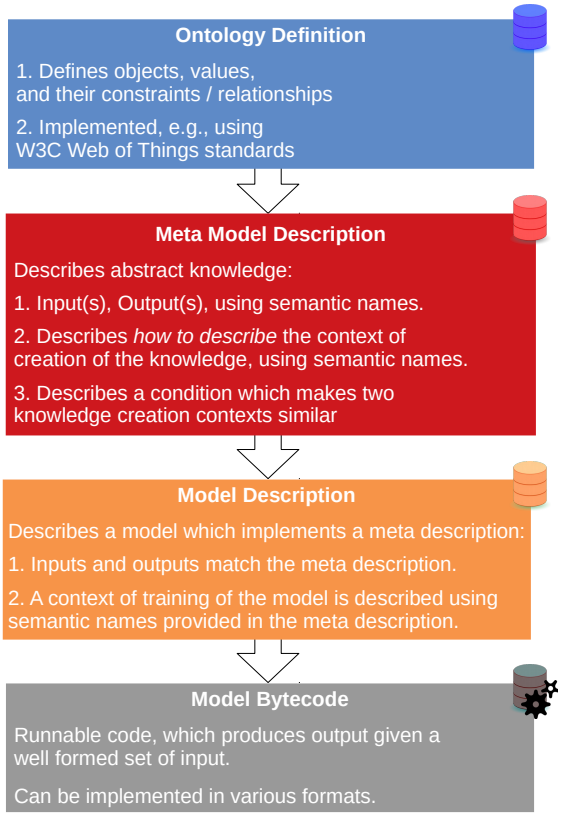


Figure 6.27: Components of the VKN Implementation

contains the `rd_exit_proba_estimator` meta description. On the other hand, the KB module of the infrastructure units is populated with the same meta description. What is more, depending on the implemented scenario, i.e., *context-agnostic*, *context-aware (without caching)*, or *context-aware*, as described in Figure 6.25, the model KB of infrastructure nodes may contain one or several descriptions of models, trained on existing roundabouts, which implement the meta description, as well as the corresponding bytecodes in the Bytecode KB.

As such, in the next section, we contribute the definition of context reasoning by an infrastructure nodes to select the known model which is the most adapted to a specific context, when replying to knowledge creation requests issued by vehicles.

Knowledge Processing Module

We define a knowledge processing module which is responsible for understanding and treating incoming knowledge networking requests as introduced in Chapter 4. Knowledge requests may be formulated to query a KB module, potentially located in a remote node. Requests may be issued to query the different databases of the implemented KB as introduced in Section 6.2.2, and as illustrated in the left-hand side of Figure 6.26:

- *Knowledge Semantic Discovery Requests*: Knowledge discovery requests aim at querying a potentially remote KB to discover semantic descriptions of knowl-

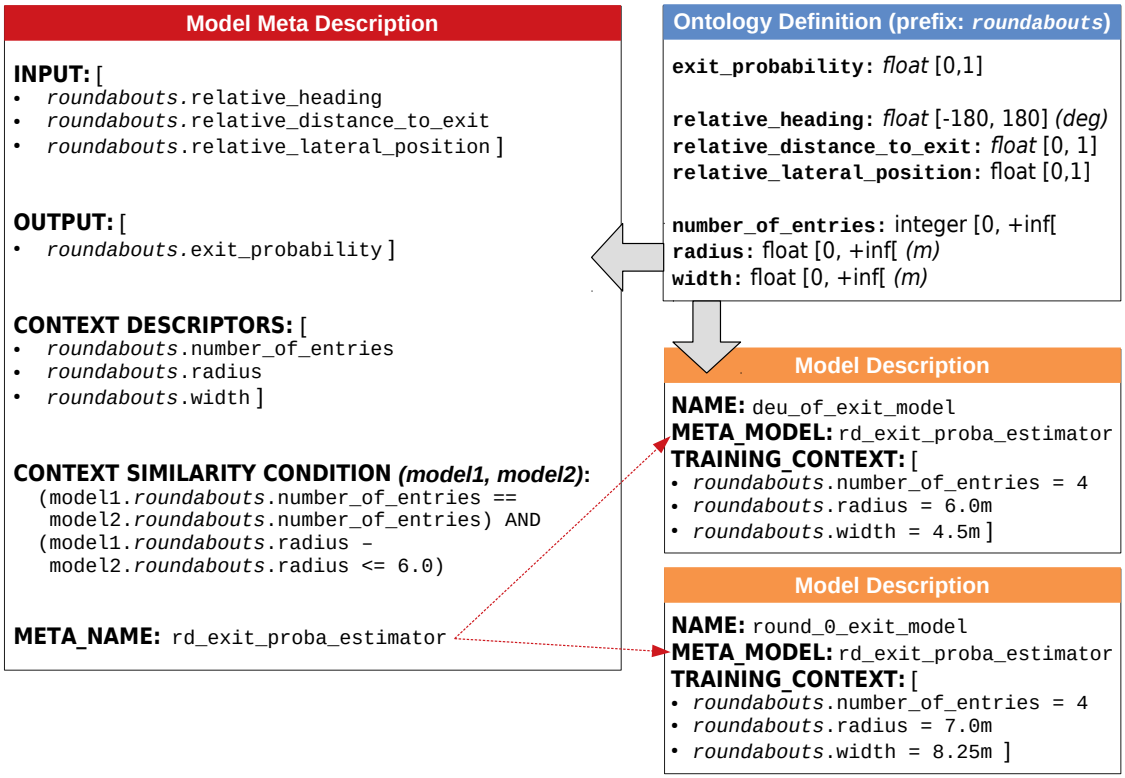


Figure 6.28: Implemented Semantic Description of Exit Probability Models

edge, which might in turn be used for knowledge retrieval requests.

  - *Meta Requests*: Meta requests may be issued by nodes to other nodes to discover meta descriptions matching specific conditions of input, output, or context. They are treated by the *Meta Semantic Matching Unit*, as shown in Figure 6.26. For example, a request might be issued to discover all the known meta descriptions which involve models that output the `roundabouts.exit_probability` content. In this use case, the response would list the `rd_exit_proba_estimator` meta description.
  - *Model Requests*: Model requests may be issued to discover a list of named models which are known by a node to implement a given meta model, in a given context. They are treated by the *Model Semantic Matching Unit* shown in Figure 6.26. For example, a node featuring a KB populated as exemplified in Figure 6.28, would reply to a request to discover models implementing the `rd_exit_proba_estimator` meta description in a context of 4-entry roundabouts with the matching `deu_of_exit_model` and `round_0_exit_model` model descriptions.

- *Knowledge Retrieval Requests*: Knowledge retrieval requests may be issued to a node to either (i) request for a copy of the bytecode of a named model, or (ii) directly request for the remote application of a named model, given a specified well-formed set of input. The *Knowledge Remote Application Unit*, as shown in Figure 6.26, is responsible for treating knowledge retrieval requests.

  - *Model Application Requests*: Model application requests might be sent to request for the application of a named model, provided with a set of

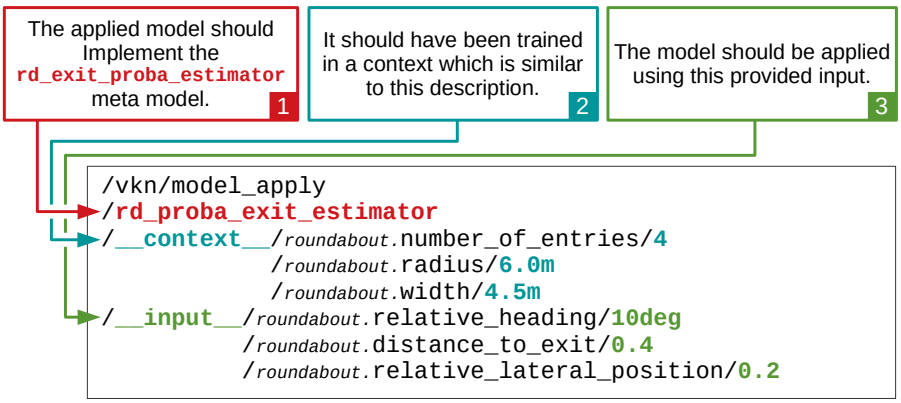


Figure 6.29: Naming Scheme for the implementation of VKN over NDN

well-formed input, the output is then produced by the *Knowledge Remote Application Unit*, e.g., the exit probability in this case, and returned to the requesting node.

– *Meta Model Application Requests*: Finally, as illustrated in Figure 6.26, instead of first performing a model knowledge semantic discovery request to get a list of named models matching a meta description in a specific context, and in turn request the remote application of one of the matching models, a node may directly request for the application of a meta model in a given context, with given input.

Figure 6.29 illustrates the last point, i.e., an example request for the creation of an exit probability value directly using a known meta description, given a set of input and a desired context of application. The request is sent from a vehicle with knowledge of the `rd_exit_proba_estimator` meta description. Then, the input values sensed from an incoming vehicle, from which the exit probability computation is requested, are attached. Finally, the vehicle includes a description of the context in which the knowledge should be created, i.e., the geometric characteristics of the crossed roundabout which are listed in the `rd_exit_proba_estimator` meta description. In turn, the infrastructure node which receives the request: (i) Looks up in its model KB to find a model which has been trained in a context which is similar to the requested context, (ii) creates the exit probability knowledge by applying that model, and (iii) returns it to the requesting vehicle. This procedure is a contribution which we define in Algorithm 5.

What is more, Figure 6.29 describes the implementation of *meta model application requests*, as part of the ns-3 simulation. Communications are implemented using NDN messages sent over a wireless channel between entering vehicles and the infrastructure nodes in the vicinity of the roundabout. As such, we encode *meta model application requests* in NDN interest packets with a hierarchical name, as described in Chapter 2. As illustrated by Figure 6.29, the interest name starts with the `/vkn/model_apply` prefix to indicate that the following will be a remote model application query interest. Then, two keywords are integrated as part of the hierarchical name. They are identified by the two underscore characters which start and end them. Specifically, `__context__` and `__input__`, respectively, indicate the start of the definition of (i) the context, and (ii) the input values associated with

Algorithm 5 Knowledge Creation Request Processing

```
1: KB_meta ← MetaKB()                                              ▷ The Meta KB.
2: KB_model ← ModelKB()                                            ▷ The Model KB.
3:
4:          ▷ Treat a 'knowledge as a service' exit probability knowledge creation NDN interest.
5: procedure treat_request(interest_name)
6:     (meta_model, inputs, context) ← parse_ndn_name(interest_name)
7:
8:     matching_models ← list()
9:     matching_models_predictions ← list()
10:
11:    for each model in KB_model do
12:        if model.meta_model = meta_model and meta_model.is_similar(model.context,
   context) then
13:        ▷ A model was found which implements the right meta model in the requested context
14:            matching_models.add(model)
15:            matching_models_predictions.add(model.apply_to(inputs))
16:        end if
17:    end for
18:
19:    if not matching_models.empty() then
20:                    ▷ In this case, we implement ensemble voting, the predictions are averaged
21:        return matching_models_predictions.average()
22:    else
23:        return no matching model
24:     ▷ In the context-agnostic knowledge networking case, the knowledge creation falls back to
   selecting a non context relevant model
25:    end if
26: end procedure
```

Table 6.4: Simulation Parameters for the Evaluation of VKN Context-Aware Knowledge Networking

Parameter	**Value**
Considered Roundabouts	Table 6.1, except USA_FT and CHN_LN
Infrastructure Nodes	1 (Local MEC) + 50 (Remote Static Nodes)
Node Placement Area	$200 \times 200m^2$
Protocol stack	IEEE 802.11p & IEEE 1609.4 & NDN
Three Log Distance Model	Distance=(1, 200, 500)m, Exponents=(1.9, 3.8, 3.8), Reference Loss=46.67dB
Nakagami Model	Distance=(80, 200)m, Exponents=(1.5, 0.75, 0.75)
NDN Hop Limit	10

the meta description. After the context or input keyword, the set of $(key, value)$ elements are separated by slashes as illustrated in Figure 6.29.

The interest is forwarded from entering vehicles, i.e., *knowledge consumers*, to the infrastructure nodes which are able to produce the requested knowledge, i.e., *knowledge producers* through multicast and multi-hop wireless communications. Upon reception of an interest of this form, the *knowledge processing module* of the knowledge producer is able to parse the hierarchical name to extract the requested meta model, context, and input values to create the requested knowledge. The model which is selected to be applied should (i) implement the meta description, (ii) have a context of training which is similar to the requested context of application, according to the similarity condition defined in the meta description, and (iii) be applied to the provided input. In turn, the produced exit probability value is returned to the original emitter of the NDN interest packet.

6.2.3 Evaluation Setup

In this section, we describe the formal simulation setup for VKN exit probability knowledge networking, based on the overview introduced in Section 6.2.1 and the description of the mechanisms of the VKN implementation in Section 6.2.2.

We run ns-3 simulations of vehicles crossing a roundabout and issuing remote knowledge creation requests for the exit probability of potential conflicting vehicles to infrastructure nodes in the vicinity of the roundabout. We aim to show that, through a semantic description of knowledge and VKN implementation, as detailed in Section 6.2.2, knowledge which is pertinent to the current context can be applied and lead to improved performance compared with a traditional approach where the context of usage of models is not taken into account. Table 6.4 summarizes the parameters of the simulations.

Topology

Each simulation is associated with one roundabout R, extracted from the list in Table 6.1, and one recording of vehicle tracks from that roundabout R. Two types of nodes are introduced:

- Infrastructure nodes or static connected objects: Specifically, a local MEC unit is placed at the center of the roundabout R. What is more, 50 additional static nodes are randomly placed in a square area of $200m$ side centered on the roundabout, such that no infrastructure node is located more than $30m$ away from all the others.

- Vehicles which replicate the mobility of the considered recording of vehicle tracks. As such, in a first step, the mobility of vehicles as it is described in RounD or Interaction dataset files is converted to the *ns-2 mobility trace* format as defined in [211], which can be understood by the ns-3 simulator.

Protocol Stack

The following protocol stack is installed on both the vehicles and the infrastructure units:

Physical & MAC layer The low layers of the networking stack up to the MAC layer follow the IEEE 802.11p protocol which defines wireless access in vehicular contexts [36], as summarized in Section 2.1. Specifically, the physical layer uses an Orthogonal Frequency-Division Multiplexing (OFDM) modulation in a 10MHz frequency band of the licensed 5.9 GHz band of ETSI ITS. In turn, the MAC layer implements IEEE 802.11p with the WAVE IEEE 1609.4 extension [212].

What is more, a propagation loss and a fading model are added to the physical IEEE 802.11p channel, i.e., respectively, the *Three Log Distance* propagation loss model, and the *Nakagami* fading model with default ns-3 parameters. Namely, the parameters of the *Three Log Distance* model are (i) Distance=$(1, 200, 500)m$, (ii) Exponents=$(1.9, 3.8, 3.8)$, and (iii) Reference Loss=$46.67dB$. The parameters of the Nakagami model are (i) Distance=$(80, 200)m$ and (ii) Exponents=$(1.5, 0.75, 0.75)$.

Networking Layer The networking layer implements the NDN protocol, using the ndnSIM 2.8 library [210]. On the one hand, entering vehicles which issue interests for the creation of exit probability knowledge of incoming vehicles are *knowledge consumers*. They produce interests for exit probability knowledge following the naming scheme illustrated in Figure 6.29.

On the other hand, depending on the considered scenario listed in Figure 6.25, one or several infrastructure nodes are defined as *knowledge producers* for exit probability values:

1. In the *context-agnostic (baseline)* approach, only the local MEC unit in the center of the roundabout is defined as a knowledge producer for exit probability values. However, the knowledge is produced without considering the context of usage of models, i.e., the model KB of the local MEC unit is populated with exit probability models which have not been trained in a relevant context for the considered roundabout R.

2. In the *context-aware, without caching* approach, 5 infrastructure units distinct from the central MEC unit are randomly selected and defined as knowledge producers. What is more, their model KB is populated with the known exit probability knowledge models trained in a similar context than that of the crossed roundabout R. As the knowledge producers are spread in the vicinity of the roundabout, multi-hop forwarding of exit probability knowledge creation requests may be required.

3. In the *context-aware* approach, optimized with context-aware knowledge caching, the known exit probability knowledge models trained in a similar context than that of the crossed roundabout R are directly cached in the central MEC unit, which is defined as the only knowledge producer.

Knowledge creation interests are routed from consumers to producers through a broadcast and multihop approach with a limit of 10 hops. In the context-agnostic and context-aware optimized with caching scenarios, the knowledge is cached in the central MEC unit close to the vehicles. As such, one hop is sufficient to retrieve the exit probability knowledge. On the contrary, multiple hops are typically required in the *context-aware, without caching* scenario.

Application Layer On top of the networking stack which is installed on each node, applications are defined, respectively, for vehicles to request the creation of exit probability knowledge and for the infrastructure knowledge producers to treat incoming requests.

As illustrated by Figure 6.1, vehicles which are entering the roundabout are detected through entry areas, as illustrated in Figure 6.12. When entering the roundabout at an entry, each vehicle v_{entry} senses whether an incoming vehicle is approaching the exit located directly before that entry. If no potential conflicting vehicle is detected, no further procedure is performed. If several potential conflicting vehicles are detected by v_{entry}, the closest vehicle in straight-line distance to v_{entry} is targeted as the potential conflicting vehicle v_{target}.

In turn, kinematics data about v_{target} are sensed by v_{entry}, i.e., the three input values defined by the `rd_exit_proba_estimator` meta description: `relative_heading`, `relative_distance_to_exit`, and `relative_lateral_position`. The sensing is simulated by extracting the data directly from the RounD or Interaction vehicle tracks. What is more, we consider the vehicle v_{entry} to possess contextual information about the roundabout R being crossed, i.e., `number_of_entries`, `radius`, `width`, e.g., extracted from local map data.

Finally, v_{entry} issues an interest to request the creation of the exit probability of v_{target}, provided (i) the input kinematics data of v_{target}, and (ii) the contextual data about the roundabout, such that the knowledge can be produced using an exit probability model trained in a similar context. The format of the interest message is shown in Figure 6.29. The interest is routed through multicast and potentially multihop communications to an infrastructure knowledge producer which implements the procedure described by Algorithm 5 to produce the requested knowledge.

Upon reception of the exit probability knowledge created by a remote infrastructure knowledge producer, the entering vehicle stores the value in its local memory for scoring the results at the end of the simulation.

Benchmark & Scoring

For each of the considered scenarios defined in Figure 6.25, we evaluate both (i) the accuracy of the produced exit probability knowledge, and (ii) networking performance metrics, such as the overhead and delay linked to knowledge retrieval.

Simulations are run for each recording of each roundabout R listed in Table 6.1 which features at least one similar roundabout according to the similarity condition of Equation 6.4. This excludes the `USA_FT` and `CHN_LN` roundabouts.

Accuracy At the end of each simulation, the probability predictions received by vehicles are scored to evaluate their accuracy. For each vehicle v_{enter} having requested and received an estimation of the probability p of exit of a sensed vehicle v_{target} from an infrastructure knowledge producer, the probability p is compared with the observed behavior of v_{target}. Namely, iff $p > 0.5$, we consider that the prediction is that v_{target} will exit, and this is matched with whether v_{target} did exit. Thus, for all predictions associated with a simulation, a value of the predicted as well as the observed behavior of the vehicle is available. In turn, accuracy scores can be computed for the set of predictions, i.e., accuracy, F1-score, or precision, as introduced in Section 6.1.4. Finally, the accuracy scores are compared for each recording of a roundabout between the baseline and proposed approaches listed in Figure 6.25.

Networking Metrics What is more, for each simulation, the following networking metrics are collected and compared for each of the scenarios, baseline and proposed, listed in Figure 6.25:

- *Delay*: The average delay between the sending of remote knowledge creation interests and the reception of the computed exit probability value.
- *Number of hops*: The average number of hops to reach a knowledge producer node.
- *Overhead*: The average overhead associated with a complete simulation, replaying the vehicular mobility of a specific track recording.

For each metric, 95% confidence intervals are computed by running 20 simulations for each roundabout recording and scenario, with different random samplings of (i) infrastructure unit locations, as well as (ii) selection of infrastructure knowledge producers, when applicable, i.e., in the context-aware (without caching) approach.

6.2.4 Results & Discussion

In this section, we describe and discuss the obtained results, both in terms of the accuracy of the produced knowledge and the networking metric performances. Specifically, the proposed VKN-supported context-aware knowledge networking approach is compared with the baseline context-agnostic approach.

Knowledge Accuracy

Figures 6.30 and 6.31 illustrate, respectively, the obtained results in terms of knowledge accuracy, comparing the baseline context-agnostic and the proposed context-aware with caching scenarios, for simulations replaying the mobility of recordings extracted from the `DEU_OF` and `RounD_0` roundabouts. The blue-colored dots represent the F1 score of the predictions provided by the infrastructure units through *context-aware* knowledge networking. In this case, the knowledge producer performs semantic reasoning to apply an ensemble voting combination of the most contextually-relevant models.

The red-colored dots with error bars represent the F1 score of predictions obtained for the context-agnostic approach, in which the knowledge producer uses an exit probability model M_i which was trained in a non-similar context. One simulation is run for each model M_i. In turn, 95% confidence intervals are computed based on the results of each simulation. The recordings in the x-axis are sorted from the lowest to the greatest F1-score difference between the proposed context-aware and the baseline context-agnostic approaches. What is more, only the recordings which feature more than 25 situations of probability prediction are represented, to avoid computing global accuracy metrics on extremely small samples of data.

Similarly, Figure 6.32 shows the precision metric of the exit probability predictions obtained through the context-aware model selection for USA_SR, compared to the random selection of a model without a similar context.

To begin with, we observe a significant increase in accuracy linked to VKN-supported context-aware knowledge networking, compared to the baseline approach which does not take the context of knowledge application into account, for the `DEU_OF` and `RounD_0` roundabouts. What is more, albeit not illustrated by a figure as they feature only one recording, the same pattern is observed for the `RounD_1` and `RounD_2` roundabouts. The exit probability knowledge which was obtained after context-aware model selection by the knowledge producer outperforms the knowledge obtained with approaches which select a random model, which has not been trained in the right context. While a random model selection approach might select a model which is adapted to the considered context by accident or by chance, the large confidence intervals shown in Figures 6.30 and 6.31, and similarly for `RounD_1` and `RounD_2`, would incite vehicles to take conservative interpretations of the obtained knowledge, i.e., assume that the accuracy is at the lowest possible level, and avoid entering in the presence of any incoming vehicle.

Then, the accuracy and F1 scores obtained for the `USA_SR` roundabout remained stable when using context-similar models compared with using other models. As

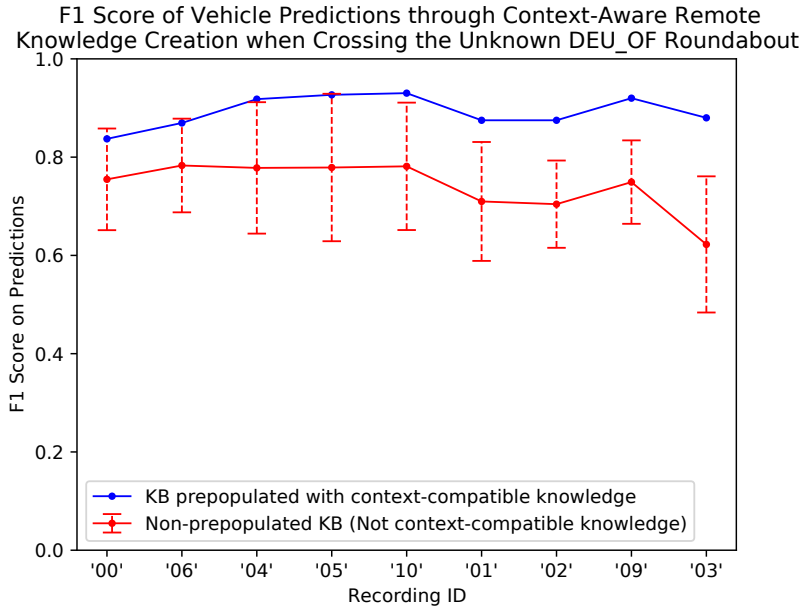


Figure 6.30: F1 Score on DEU_OF Exit Predictions using Context Aware Knowledge Networking

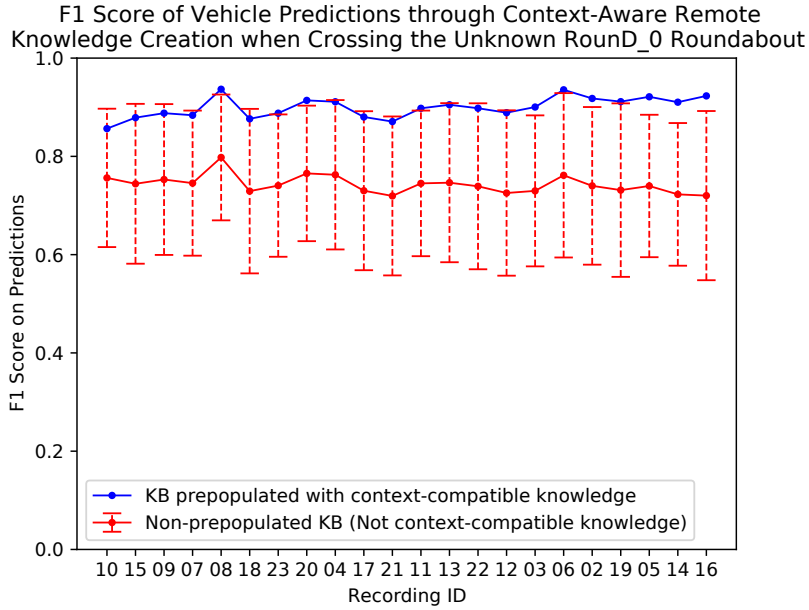


Figure 6.31: F1 Score on RounD_0 Exit Predictions using Context Aware Knowledge Networking

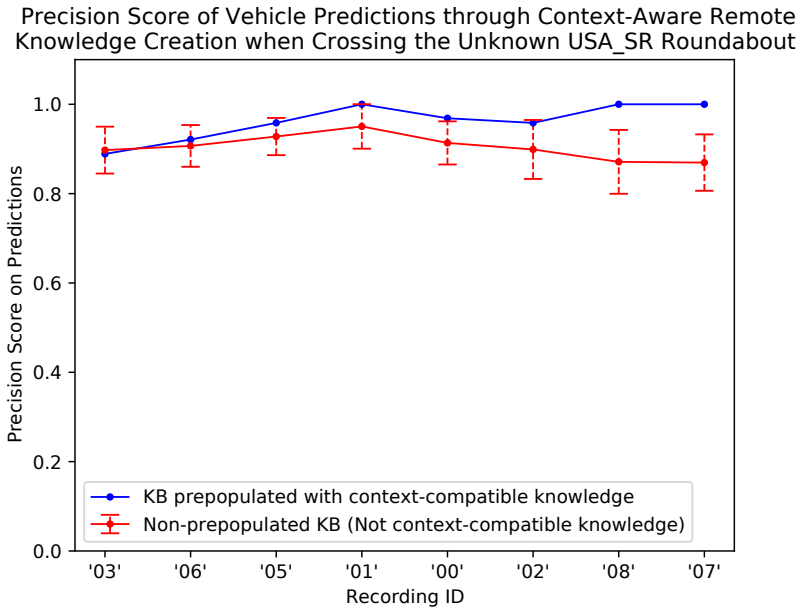


Figure 6.32: Precision Score on USA_SR Exit Predictions using Context Aware Knowledge Networking

shown in the previous section, and specifically in Figure 6.18. This could be explained by the used similarity condition not being strict enough. Nonetheless, the context-aware model selection approach improved the precision of the created knowledge, as illustrated by Figure 6.32. Moreover, the `USA_EP` roundabout follows the same pattern. This means that the context-aware approach is more robust to false positives, i.e., predicting the exit of a vehicle which stays in the roundabout, which poses a safety risk as entering vehicles in turn wrongly assume they can enter. As such, the context-aware approach nonetheless improves a key metric of the exit probability risk model performance.

Networking Performance

In turn, we evaluate the performance of exit probability knowledge distribution in terms of delay, number of hops and overhead, for each of the baseline and proposed approaches listed in Figure 6.25. As the networking performance metrics linked with the retrieval of a value of exit probability knowledge are independent from the used exit probability model, we focus, for the evaluation, on a recording taken from the `DEU_OF` roundabout, i.e., the recording of identifier '00'.

Figure 6.33 illustrates the obtained results and compares the proposed approaches with the baseline. For each scenario, the average accuracy of the produced knowledge, as well as the average delay to retrieve it, is shown, respectively, in the y-axis and x-axis, with 95% confidence intervals. What is more, the average number of hops per knowledge request, and average overhead in a simulation, are represented by the color and size of each point.

To begin with, we notice that the accuracy of the knowledge produced in the baseline context-agnostic approach is significantly lower than that of the knowledge

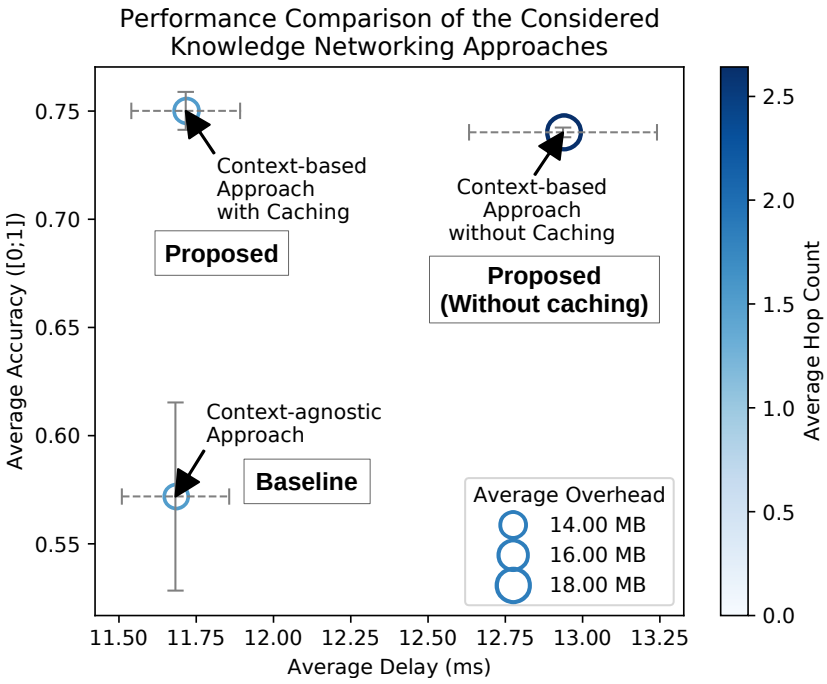


Figure 6.33: Networking Performance Evaluation

produced in the proposed context-aware approaches, as shown in Figure 6.30. Yet, in the context-aware approach without caching, the gain in accuracy is compensated by a loss in terms of networking performance, with a higher delay, number of hops, and overhead in average.

In turn, in the proposed *context-aware networking* approach, optimized with caching, knowledge in the right context is directly cached close to the roundabouts in which it is relevant. In turn, a significantly increased accuracy can be achieved while maintaining values of delay, number of hops and overhead which are equivalent to the baseline approach.

The obtained results illustrate the potential and impact of context-based knowledge networking both in terms of accuracy of the produced knowledge and of networking efficiency metrics. Furthermore, they indicate the key role and potential of a vehicular knowledge networking framework to open knowledge networking to all nodes of vehicular networks while maintaining the accuracy of knowledge despite the variety of driving contexts. A key to achieve this is to define precise knowledge description semantics, in turn supporting knowledge caching and dissemination operations.

In this section, we contributed a semantic description of the roundabout exit probability models which we defined in Chapters 5 and 6. In turn, we showed the need for a specific knowledge module architecture as part of the VKN framework, which we defined. In turn, we contributed a generic algorithm to perform context-aware model selection to provide *knowledge as a service* operations. In this work, we focused on the networking aspect of the exit probability retrieval use case, for a self-driving vehicle which wishes to enter a roundabout. Yet, the mobility of vehicles was not actually modified as a consequence of the obtained exit probabilities. As future work, the algorithms of knowledge storage and context-aware creation could

be used as part of a real driving-assistance use case, where the vehicles which enter a roundabout would take educated decisions on whether to enter, based on the dissemination of exit probability knowledge supported by the VKN framework.

Chapter 7

Conclusion and Perspectives

In this chapter, we summarize the key conclusions obtained through this doctoral work. Furthermore, the described conclusions are used to open perspectives for future works to allow and improve the support of knowledge networking in vehicular networks. Lastly, the publications associated with this doctoral work are listed.

7.1 Conclusion

Knowledge networking, as opposed to information networking, is emerging as a paradigm to allow the mutual understanding and distribution of abstract knowledge in vehicular networks. Knowledge can be defined as an abstract ability to reason about and understand the environment in a specific context, and can be created through various AI algorithms, including ML. So far, numerous mechanisms of information sharing and standards have been defined to uniquely name key vehicular content, such as safety notifications through CAM or DENM messages. Such mechanisms are efficient to exchange the output of knowledge models, but not the knowledge models themselves. For example, let us consider a model to produce the exit probability of a vehicle in a roundabout given its kinematics data, as introduced in Chapter 5. While the computed value of exit probability for a vehicle produced by the model can be exchanged using existing information-sharing standards, the distribution of the exit probability computation knowledge model itself faces specific challenges. Namely, due to the strong contextual meaning of knowledge, which has been trained to be used in a specific context, knowledge networking mechanisms should integrate naming mechanisms which convey the contextual meaning of knowledge. A requirement for knowledge networking is that nodes of the vehicular networks share a common understanding of the context in which a knowledge model can be applied or further trained. In turn, knowledge can be reused and not independently retrained by multiple entities.

As such, in this doctoral work, we performed several sequential tasks to:

1. Frame the requirements for knowledge networking, as opposed to information networking.

2. Identify to what extent and in which areas the performance of vehicular applications could benefit from vehicular knowledge networking.

3. Define a conceptual framework to implement knowledge networking in vehicular networks, based on this understanding.

4. Evaluate the improved performances linked to vehicular knowledge networking on a roundabout risk knowledge networking case study.

In the next paragraphs, we describe and discuss the key conclusions linked to each of these points.

- A key requirement which is an enabler for knowledge networking in vehicular networks is the semantic description of knowledge. Content description mechanisms are already in place for information, e.g., CAM messages to describe vehicle kinematics information, or DENM to describe punctual safety-related vehicular events. Yet, the description of knowledge is more complex and requires semantic description which goes beyond the definition of named information items. Namely, knowledge is a highly contextual content, and a knowledge model, e.g., trained using ML algorithms, has been trained using data extracted in a specific context. As such, the definition of means to describe the context in which a knowledge model can be applied is critical to ensure its accuracy, as exemplified in the roundabout exit probability knowledge networking case study of Chapter 6.

  What is more, knowledge description standards should allow a listing of the interface, i.e., input and output items of a model. In turn, nodes which did not participate in the design of the knowledge model may use it. Semantic Web and WoT technologies are candidate semantic description standards which allow the description of the interface, as well as context of training and application of a knowledge model.

- Knowledge networking involves a larger set of operations than information networking. Like information, the machine code of knowledge models can be transferred directly using existing V2X vehicular exchange channels. Furthermore, additional networking operations are relevant to knowledge, such as:

  - The remote application of a knowledge model by a distant node, without retrieving the model.
  - The cooperative training of knowledge by distributed nodes, e.g., using FL algorithms.

  On the one hand, one area of performance improvement which can be reached by knowledge networking mechanisms is model training and application accuracy. In Section 4.3, we demonstrate an improvement of the speed of training of a FL model in vehicular contexts, as well as an increased accuracy of the trained model. Our approach uses a context-aware algorithm which selects the training nodes which are most likely to possess the right type of data and context to participate in training. Moreover, in Chapter 6, we show improvements in the accuracy of knowledge remotely created in infrastructure units

in a case study of exit probability knowledge estimation. Through semantic context reasoning, the infrastructure units select the knowledge models which have been trained in similar contexts than the requested context of application. In turn, the accuracy of exit probability predictions is significantly increased compared to a traditional approach which does not take context into account when sharing knowledge, while maintaining an equivalent overhead and delay for knowledge retrieval.

On the other hand, in Section 4.2, we show that the operation of remote knowledge creation can reduce overhead in vehicular networks compared to an approach which directly transfers the machine code of the considered knowledge model, starting from a specific size threshold.

- We defined a conceptual framework for knowledge networking in vehicular networks called *Vehicular Knowledge Networking*, in Chapter 4. It consists of three main units: Knowledge storage, knowledge queries, and knowledge processing, all supported by knowledge description semantics, which are a key aspect of the framework.
  - The knowledge base unit can contain both the machine code of knowledge models and semantic descriptions of their interface and relevant context of application.
  - Knowledge queries can be issued by a node of the vehicular network to itself or to a remote node. Knowledge queries aim to address the set of operations related to knowledge networking, as described in the previous point. As such, queries should be defined for (i) the lookup of existing models, based on interface or context filters, (ii) requests for a copy of a model machine code, or the direct remote application of that model, provided a set of input, and (iii) requests to further train a model in a distributed approach.
  - The knowledge processing unit is responsible to answer to knowledge queries by using the knowledge base of a node to select the best model to apply or train based on the context in which the knowledge is requested. In Section 6.2.2, we describe and evaluate an implementation of a knowledge processing unit for context-aware remote knowledge applications.

  In Section 6.2.2, we perform an implementation of VKN over NDN for a case of roundabout exit probability knowledge networking. A key conclusion from this point is that knowledge discovery and remote knowledge application operations can be implemented as applications running as an overlay over existing content sharing standards such as information-centric networking, and specifically NDN, rather than needing specific networking layer protocols. Namely, provided that the set of inputs of the considered knowledge model can be conveniently encoded as text, the contextual metadata and model description needed to request the remote application of this knowledge model can be encoded and conveyed as part of a NDN content name.

- We find that the integration of semantics to describe the context of application of a knowledge model in knowledge networking operations is beneficial for the accuracy of the produced knowledge.

In Section 6.1, we performed a study to determine the semantic objects which are most relevant to describe the context of training of a model which estimates the probability of a vehicle to exit at the next available exit in a roundabout. Eight distinct models were trained using training data, i.e., vehicle tracks, extracted from eight distinct roundabouts. We find that low differences in terms of (i) the number of entries, (ii) the radius, and (iii) the width of two roundabouts lead to a high mutual information of the associated trained exit probability models. In turn, we observe a higher accuracy when applying an exit probability estimation model in a roundabout, which was not used to train the model, if the considered roundabout has low *(entries, width, radius)* differences with the roundabout that was used to train the model. Thus, we identify the three semantic objects of roundabout geometry *(entries, width, radius)* as relevant to describe the context of training of an exit probability model.

Then, in Section 6.1.4, we showed performance improvements in knowledge networking though context-aware remote knowledge applications. Based on the *(entries, width, radius)* semantic contextual parameters, as well as a condition to determine whether the context associated with two distinct roundabouts is similar, the node which performs knowledge creation upon requests of vehicles crossing an unknown roundabout, i.e., a roundabout for which no exit probability model has been trained, is able to select and apply the existing models which have been trained in a similar context than the crossed roundabout, and in turn improve the accuracy of the produced exit probability content.

What is more, in the same Section 6.1.4, accuracy improvements were obtained for exit probability model training, when a model for an unseen roundabout is trained using a fraction of training data extracted from roundabouts featuring a similar context. In Section 4.3, accuracy and training speed improvements were obtained in a cooperative model training study, when training nodes are communicating information about their current context of training, which is used by the FL training coordinator to select appropriate training nodes.

As a key conclusion for this point, we argue that similar context analysis could be performed by model authors for models which strongly depend on a context, e.g., object recognition in various locations and weather conditions. In turn, model-wise context description semantics could be defined, and allow increased performances of knowledge networking in vehicular networks.

7.2 Perspectives

To begin with, we describe perspectives which are closely related to the works and results presented as part of this doctoral work. Then, we conclude by discussing more broad and generic perspectives for the field of vehicular networking, which were opened by this dissertation.

7.2.1 Specific Perspectives

The following perspectives were opened by the specific results obtained through each work item:

- In Section 5.1, we evaluated the impact of strictly routing CAVs to avoid areas which were flagged as *risky* for their autonomous driving units. The results indicated an increase in the queue length, as well as of pollutant emissions throughout the simulations. This simulation setup was intentionally extreme in the strict risk area avoidance, to show the need for a finer routing based on a dynamic risk knowledge. As future work, these results could be put in perspective with a simulation where CAVs are routed to avoid risky areas, based on a dynamic assessment of risk in various areas of a city, i.e., using the roundabout risk knowledge which was defined in Chapter 5.

- In Section 6.1, multiple roundabouts were considered to train multiple vehicle exit probability estimation models, and estimate the key semantic items to describe the similarity between the contexts of training of two roundabouts. As future work, additional roundabout tracks data from novel datasets could be used to further refine the content which is required to describe the context of a roundabout, as well as evaluate the obtained results on additional data.

- In Section 6.2, we showed an increased accuracy for knowledge networking and the remote application of knowledge models when they are applied in the right context. Namely, vehicles crossing a roundabout were able to receive the exit probabilities of incoming vehicles with increased accuracy or precision, compared to a traditional case, which does not take the context of creation of knowledge into account. So far, the probability values are received by the vehicles which are about to enter a roundabout as a proof of concept, but the mobility of entering vehicles is not altered to adapt to the obtained exit probability values. As a future work item, the exit probability knowledge received by entering vehicles could be integrated with an advanced driver assistance system and adapt their mobility to the probability of conflict with an incoming vehicle.

- A key conclusion of this doctoral work is the importance of the definition of a mutually understandable description of knowledge. Through descriptions of the context of use and training of a piece of knowledge, increased accuracy of knowledge networking has been showed, both for remote model application in Sections 6.1.4 and 6.2 and for cooperative model training in Sections 4.3 and 6.1.4. In order to allow the generalization of knowledge networking in vehicular networks, we argue that model makers should proceed to two extra steps when training models. On the one hand, a semantic description of the interface should be defined in a mutually understandable format, such that other nodes can use the knowledge. On the other hand, an extra analysis to identify and describe the context of application and training of a model should be defined to allow its application or training in the right context and, in turn, reach an increased accuracy of the content produced by the model.

7.2.2 Generic Perspectives

Furthermore, the analysis, results and impact obtained and presented in this doctoral work allow us to open more generic and broad perspectives for future vehicular networks:

- The security of knowledge networking in vehicular networks is a key aspect for real world implementations. For example, if a knowledge model is created by one agent, further trained or composed with existing knowledge from a second agent, and finally used by a third agent, mechanisms should be implemented to ensure the authentication of nodes and the traceability of the 'history' of a piece of knowledge, to avoid the distribution of falsified knowledge. While it is complex to detect forged or false information, it is even more so for knowledge, which can take many forms and whose internal functioning can be difficult to interpret, e.g., with ANNs. As such, a future perspective for vehicular knowledge networking is the integration of distributed security to the distributed knowledge creation, storage, and dissemination aspects.
- Vehicular knowledge networking could have a strong impact on the distribution of map data. Namely, an extra mapping layer could be defined to encode information about the context and driving conditions associated with each area, both statically, e.g., it is a 4-entry roundabout, and dynamically, e.g., clear weather. In turn, such map data could support knowledge caching operations, which cache knowledge in areas which feature a relevant context for it. This implies that local actors, e.g., roundabout builders, publish detailed contextual information about each area, e.g., the complete semantic description of the built roundabout. As such, this requires cooperation with a large spectrum of entities, and incentives might be required to encourage the definition of complex contextual map data by local actors.
- In turn, the financial aspect is potentially a key perspective for vehicular knowledge networking. Like mining cryptocurrencies, training and providing knowledge as a service is a time and resource-consuming process, which could be encouraged through monetary compensations for vehicular nodes which provide knowledge services. For example, FedCoin [156], as considered in the FL incentive approaches of Table 3.1, is a cryptocurrency to compensate nodes participating in FL. This approach could be extended to other forms of knowledge networking, such as knowledge as a service.

The Internet has been a first revolution in terms of the universal access to information, which brought a set of challenges, but ultimately improved user freedom. We believe the networking of knowledge to be the next step. In current approaches, computerized knowledge is the property of few entities, training complex DL models on proprietary and centralized data centers. In turn, an Internet of knowledge for vehicular networks would allow the democratization of the access and the cooperative evolution of knowledge. Like human drivers build the knowledge of their driving skills from initial education and subsequent continuous learning, future self-driving vehicles could adapt their driving behavior based on a personalization of existing knowledge to the current context, rather than following a preset driving algorithm.

Appendix A

List of Publications

A.1 Lead Author

- Duncan Deveaux, Takamasa Higuchi, Seyhan Uçar, Jérôme Härri, and Onur Altintas. "A Definition and Framework for Vehicular Knowledge Networking: An Application of Knowledge-Centric Networking". In: *IEEE Vehicular Technology Magazine* 16.2 (2021), pp. 57–67. DOI: 10.1109/MVT.2021.3066376

- Duncan Deveaux, Takamasa Higuchi, Seyhan Uçar, Chang-Heng Wang, Jérôme Härri, and Onur Altintas. "On the Orchestration of Federated Learning through Vehicular Knowledge Networking". In: *2020 IEEE Vehicular Networking Conference (VNC)*. 2020, pp. 1–8. DOI: 10.1109/VNC51378.2020.9318386

- Duncan Deveaux, Takamasa Higuchi, Seyhan Uçar, Chang-Heng Wang, Jérôme Härri, and Onur Altintas. "Extraction of Risk Knowledge from Time To Collision Variation in Roundabouts". In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021, pp. 3665–3672. DOI: 10.1109/ITSC48978.2021.9564570

- Duncan Deveaux and Jérôme Härri. *On the traffic impact of risk-aware autonomous vehicle routing strategies*. Tech. rep. Work presented at the Symposium on Management of Future Motorway and Urban Traffic Systems (MFTS 2020), Luxembourg, July 2020. EURECOM, 2021. URL: https://www.eurecom.fr/publication/6698

A.2 Secondary Author

- Seyhan Ucar, Takamasa Higuchi, Chang-Heng Wang, Duncan Deveaux, Jérôme Härri, and Onur Altintas. "Vehicular Knowledge Networking and Application to Risk Reasoning". In: *Proceedings of the Twenty-First International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*. Mobihoc '20. 2020, pp. 351–356. DOI: 10.1145/3397166.3413467

- Seyhan Uçar, Takamasa Higuchi, Chang-Heng Wang, Duncan Deveaux, Onur Altintas, and Jérôme Härri. "Vehicular Knowledge Networking and Mobility-Aware Smart Knowledge Placement". In: *IEEE Consumer Communications & Networking Conference (CNCC) 2022*. To appear. 2022

A.3 To be Submitted for Review

- Duncan Deveaux, Takamasa Higuchi, Seyhan Uçar, Jérôme Härri, Onur Altintas. *Role of context in determining transfer of risk knowledge in roundabouts*. To be submitted to the 'Transportation Research Part C: Emerging Technologies' (TR_C) journal. Preprint: `https://www.eurecom.fr/publication/6759`. 2022

- Duncan Deveaux, Takamasa Higuchi, Seyhan Uçar, Jérôme Härri, Onur Altintas. *A Knowledge Networking Approach for AI-driven Roundabout Risk Assessment*. Submitted to the 17th Wireless On-demand Network systems and Services Conference (WONS 2022). Preprint: `https://www.eurecom.fr/fr/publication/6763`. 2022

Bibliography

[1] Carlos Renato Storck and Fátima Duarte-Figueiredo. "A Survey of 5G Technology Evolution, Standards, and Infrastructure Associated With Vehicle-to-Everything Communications by Internet of Vehicles". In: *IEEE Access* 8 (2020), pp. 117593–117614. DOI: 10.1109/ACCESS.2020.3004779.

[2] Dinh C. Nguyen et al. "6G Internet of Things: A Comprehensive Survey". In: *IEEE Internet of Things Journal* (2021), pp. 1–1. DOI: 10.1109/JIOT.2021.3103320.

[3] Automotive Edge Computing Consortium (AECC). *Enabling the Connected Vehicle Market to Thrive (White Paper)*. Mar. 2021. URL: %5Curl%7Bhttps://aecc.org/wp-content/uploads/2021/03/Enabling_the_Connected_Vehicle-White_Paper_V5.pdf%7D.

[4] Hideki Shimada et al. "Implementation and Evaluation of Local Dynamic Map in Safety Driving Systems". In: *Journal of Transportation Technologies 05(02):102-112* (Jan. 2015). DOI: 10.4236/jtts.2015.52010.

[5] *ETSI EN 302 637-2 V1.4.1 : Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service*. Tech. rep. ETSI, Apr. 2019. URL: https://www.etsi.org/deliver/etsi_en/302600_302699/30263702/01.04.01_60/en_30263702v010401p.pdf.

[6] *ETSI EN 302 637-3 V1.3.1 : Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 3: Specifications of Decentralized Environmental Notification Basic Service*. Tech. rep. ETSI, Apr. 2019. URL: https://www.etsi.org/deliver/etsi_en/302600_302699/30263703/01.03.01_60/en_30263703v010301p.pdf.

[7] *ETSI TR 103 562 V2.1.1 : Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Analysis of the Collective Perception Service (CPS); Release 2*. Tech. rep. ETSI, Dec. 2019. URL: https://www.etsi.org/deliver/etsi_tr/103500_103599/103562/02.01.01_60/tr_103562v020101p.pdf.

[8] Gokulnath Thandavarayan, Miguel Sepulcre, and Javier Gozalvez. "Generation of Cooperative Perception Messages for Connected and Automated Vehi-

cles". In: *IEEE Transactions on Vehicular Technology* 69.12 (2020), pp. 16336–16341. DOI: 10.1109/TVT.2020.3036165.

[9] Ken Borgendale Andrew Banks Ed Briggs and Rahul Gupta. *MQTT Version 5.0. OASIS Standard*. Tech. rep. Mar. 2019. URL: https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html.

[10] K. Hartke Z. Shelby and C. Bormann. *The Constrained Application Protocol (CoAP)*. RFC 7252. RFC Editor, June 2014. URL: https://www.rfc-editor.org/rfc/rfc7252.txt.

[11] David Ingham Robert Godfrey and Rafael Schloming. *OASIS Advanced Message Queuing Protocol (AMQP) Version 1.0. OASIS Standard*. Tech. rep. Oct. 2012. URL: https://docs.oasis-open.org/amqp/core/v1.0/amqp-core-complete-v1.0.pdf.

[12] Boubakr Nour et al. "A survey of Internet of Things communication using ICN: A use case perspective". In: *Computer Communications* 142-143 (2019), pp. 95–123. ISSN: 0140-3664. DOI: 10.1016/j.comcom.2019.05.010.

[13] J. Chen, M. Jahanian, and K. K. Ramakrishnan. "Black ice! Using Information Centric Networks for timely vehicular safety information dissemination". In: *2017 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*. 2017, pp. 1–6. DOI: 10.1109/LANMAN.2017.7972151.

[14] Xiaopeng Zong et al. "Obstacle Avoidance for Self-Driving Vehicle with Reinforcement Learning". In: *SAE International Journal of Passenger Cars - Electronic and Electrical Systems* 11 (Jan. 2018). Technical report, pp. 28+. ISSN: 19464614.

[15] Pengwei Wang et al. "Obstacle Avoidance Path Planning Design for Autonomous Driving Vehicles Based on an Improved Artificial Potential Field Algorithm". In: *Energies* 12.12 (2019). ISSN: 1996-1073. DOI: 10.3390/en12122342. URL: https://www.mdpi.com/1996-1073/12/12/2342.

[16] Vivienne Sze et al. "Hardware for machine learning: Challenges and opportunities". In: *2017 IEEE Custom Integrated Circuits Conference (CICC)*. 2017, pp. 1–8. DOI: 10.1109/CICC.2017.7993626.

[17] Manolis Sifalakis et al. "An Information Centric Network for Computing the Distribution of Computations". In: *Proceedings of the 1st ACM Conference on Information-Centric Networking*. ACM-ICN '14. Paris, France: Association for Computing Machinery, 2014, pp. 137–146. ISBN: 9781450332064. DOI: 10.1145/2660129.2660150.

[18] Duncan Deveaux et al. "A Definition and Framework for Vehicular Knowledge Networking: An Application of Knowledge-Centric Networking". In: *IEEE Vehicular Technology Magazine* 16.2 (2021), pp. 57–67. DOI: 10.1109/MVT.2021.3066376.

[19] Duncan Deveaux et al. "On the Orchestration of Federated Learning through Vehicular Knowledge Networking". In: *2020 IEEE Vehicular Networking Conference (VNC)*. 2020, pp. 1–8. DOI: 10.1109/VNC51378.2020.9318386.

[20] Duncan Deveaux and Jérôme Härri. *On the traffic impact of risk-aware autonomous vehicle routing strategies*. Tech. rep. Work presented at the Symposium on Management of Future Motorway and Urban Traffic Systems (MFTS 2020), Luxembourg, July 2020. EURECOM, 2021. URL: https://www.eurecom.fr/publication/6698.

[21] Duncan Deveaux et al. "Extraction of Risk Knowledge from Time To Collision Variation in Roundabouts". In: *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. 2021, pp. 3665–3672. DOI: 10.1109/ITSC48978.2021.9564570.

[22] Duncan Deveaux et al. *Role of context in determining transfer of risk knowledge in roundabouts*. To be submitted to the 'Transportation Research Part C: Emerging Technologies' (TR_C) journal. Preprint: https://www.eurecom.fr/publication/6759. 2022.

[23] Duncan Deveaux et al. *A Knowledge Networking Approach for AI-driven Roundabout Risk Assessment*. Submitted to the 17th Wireless On-demand Network systems and Services Conference (WONS 2022). Preprint: https://www.eurecom.fr/fr/publication/6763. 2022.

[24] CAR 2 CAR Communication Consortium. *CAR 2 CAR Communication Consortium Manifesto: Overview of the C2C-CC System*. Tech. rep. Aug. 2007. URL: https://www.car-2-car.org/fileadmin/documents/General_Documents/C2C-CC_Manifesto_Aug_2007.pdf.

[25] Tamer Nadeem et al. "TrafficView: Traffic Data Dissemination Using Car-to-Car Communication". In: *SIGMOBILE Mob. Comput. Commun. Rev.* 8.3 (July 2004), pp. 6–19. ISSN: 1559-1662. DOI: 10.1145/1031483.1031487.

[26] X. Yang et al. "A vehicle-to-vehicle communication protocol for cooperative collision warning". In: *The First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004.* 2004, pp. 114–123. DOI: 10.1109/MOBIQ.2004.1331717.

[27] Georgios Karagiannis et al. "Vehicular Networking: A Survey and Tutorial on Requirements, Architectures, Challenges, Standards and Solutions". In: *IEEE Communications Surveys & Tutorials* 13.4 (2011), pp. 584–616. DOI: 10.1109/SURV.2011.061411.00019.

[28] *ETSI TR 102 638 V1.1.1 : Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions*. Tech. rep. ETSI, June 2009. URL: https://www.etsi.org/deliver/etsi_tr/102600_102699/102638/01.01.01_60/tr_102638v010101p.pdf.

[29] *C-ITS Services - Use Case Roadmap*. Tech. rep. Accessed from https://www.car-2-car.org/about-c-its/. CAR 2 CAR Communication Consortium, 2020. URL: https://www.car-2-car.org/fileadmin/downloads/PDFs/roadmap/Roadmap_2020_figure.pdf.

[30] V2X Core Technical Committee. *V2X Communications Message Set Dictionary*. 2020. DOI: 10.4271/J2735_202007.

[31] *ETSI EN 302 636-5-1 V2.2.1 : Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 5: Transport Protocols; Sub-part 1: Basic Transport Protocol*. Tech. rep. ETSI, May 2019. URL: https://www.etsi.org/deliver/etsi_en/302600_302699/3026360501/02.02.01_60/en_3026360501v020201p.pdf.

[32] *ETSI EN 302 636-1 V1.2.1 : Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 1: Requirements*. Tech. rep. ETSI, Apr. 2014. URL: https://www.etsi.org/deliver/etsi_en/302600_302699/30263601/01.02.01_60/en_30263601v010201p.pdf.

[33] *ETSI EN 302 636-2 V1.2.1 : Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 2: Scenarios*. Tech. rep. ETSI, Nov. 2013. URL: https://www.etsi.org/deliver/etsi_en/302600_302699/30263602/01.02.01_60/en_30263602v010201p.pdf.

[34] *ETSI EN 302 636-6-1 V1.2.1 : Intelligent Transport Systems (ITS); Vehicular Communications; GeoNetworking; Part 6: Internet Integration; Sub-part 1: Transmission of IPv6 Packets over GeoNetworking Protocols*. Tech. rep. ETSI, May 2014. URL: https://www.etsi.org/deliver/etsi_en/302600_302699/3026360601/01.02.01_60/en_3026360601v010201p.pdf.

[35] "IEEE Standard for Wireless Access in Vehicular Environments (WAVE)–Networking". In: *IEEE Std 1609.3-2020 (Revision of IEEE Std 1609.3-2016)* (2021), pp. 1–210. DOI: 10.1109/IEEESTD.2021.9374154.

[36] "IEEE Standard for Information technology– Local and metropolitan area networks– Specific requirements– Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications Amendment 6: Wireless Access in Vehicular Environments". In: *IEEE Std 802.11p-2010 (Amendment to IEEE Std 802.11-2007 as amended by IEEE Std 802.11k-2008, IEEE Std 802.11r-2008, IEEE Std 802.11y-2008, IEEE Std 802.11n-2009, and IEEE Std 802.11w-2009)* (2010), pp. 1–51. DOI: 10.1109/IEEESTD.2010.5514475.

[37] *IEEE P802.11-TASK GROUP BD (NGV) MEETING UPDATE ; Status of Project IEEE P802.11bd*. Tech. rep. Accessed 04 October 2021. URL: https://www.ieee802.org/11/Reports/tgbd_update.htm.

[38] *ETSI TR 121 914 V14.0.0 : Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE; 5G; Release description; Release 14 (3GPP TR 21.914 version 14.0.0 Release 14)*. Tech. rep. ETSI, June 2018. URL: https://www.etsi.org/deliver/etsi_tr/121900_121999/121914/14.00.00_60/tr_121914v140000p.pdf.

[39] *ETSI TR 121 916 V16.0.1 : Digital cellular telecommunications system (Phase 2+) (GSM); Universal Mobile Telecommunications System (UMTS); LTE; 5G; Release 16 Description; Summary of Rel-16 Work Items (3GPP TR 21.916 version 16.0.1 Release 16)*. Tech. rep. ETSI, Sept. 2021. URL: https://www.etsi.org/deliver/etsi_tr/121900_121999/121916/16.00.01_60/tr_121916v160001p.pdf.

[40] *ETSI TS 122 185 V16.0.0 : LTE; Service requirements for V2X services (3GPP TS 22.185 version 16.0.0 Release 16)*. Tech. rep. ETSI, Aug. 2020. URL: https://www.etsi.org/deliver/etsi_ts/122100_122199/122185/16.00.00_60/ts_122185v160000p.pdf.

[41] George P. Corser, Huirong Fu, and Abdelnasser Banihani. "Evaluating Location Privacy in Vehicular Communications and Applications". In: *IEEE Transactions on Intelligent Transportation Systems* 17.9 (2016), pp. 2658–2667. DOI: 10.1109/TITS.2015.2506579.

[42] Miguel Sepulcre and Javier Gozalvez. "Context-aware heterogeneous V2X communications for connected vehicles". In: *Computer Networks* 136 (2018), pp. 13–21. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2018.02.024.

[43] Le Liang et al. "Deep-Learning-Based Wireless Resource Allocation With Application to Vehicular Networks". In: *Proceedings of the IEEE* 108.2 (2020), pp. 341–356. DOI: 10.1109/JPROC.2019.2957798.

[44] Xianfu Chen et al. "Learning to Entangle Radio Resources in Vehicular Communications: An Oblivious Game-Theoretic Perspective". In: *IEEE Transactions on Vehicular Technology* 68.5 (2019), pp. 4262–4274. DOI: 10.1109/TVT.2019.2907589.

[45] Farooque Hassan Kumbhar and Soo Young Shin. "Novel Vehicular Compatibility based Ad-hoc Message Routing Scheme in the Internet of Vehicles using Machine Learning". In: *IEEE Internet of Things Journal* (2021), pp. 1–1. DOI: 10.1109/JIOT.2021.3093545.

[46] Sumudu Samarakoon et al. "Distributed Federated Learning for Ultra-Reliable Low-Latency Vehicular Communications". In: *IEEE Transactions on Communications* 68.2 (2020), pp. 1146–1159. DOI: 10.1109/TCOMM.2019.2956472.

[47] Mohammad Irfan Khan et al. "Deep Learning-aided Resource Orchestration for Vehicular Safety Communication". In: *2019 Wireless Days (WD)*. 2019, pp. 1–8. DOI: 10.1109/WD.2019.8734252.

[48] Mayank K. Pal et al. "A Reinforcement Learning Approach to Jointly Adapt Vehicular Communications and Planning for Optimized Driving". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 3287–3293. DOI: 10.1109/ITSC.2018.8569484.

[49] Ahsan Mahmood et al. "The RICH Prefetching in Edge Caches for In-Order Delivery to Connected Cars". In: *IEEE Transactions on Vehicular Technology* 68.1 (2019), pp. 4–18. DOI: 10.1109/TVT.2018.2879850.

[50] Ke Zhang et al. "Digital Twin Empowered Content Caching in Social-Aware Vehicular Edge Networks". In: *IEEE Transactions on Computational Social Systems* (2021), pp. 1–13. DOI: 10.1109/TCSS.2021.3068369.

[51] Qi Qi et al. "Knowledge-Driven Service Offloading Decision for Vehicular Edge Computing: A Deep Reinforcement Learning Approach". In: *IEEE Transactions on Vehicular Technology* 68.5 (2019), pp. 4192–4203. DOI: 10.1109/TVT.2019.2894437.

[52] Zhaolong Ning et al. "Intelligent Edge Computing in Internet of Vehicles: A Joint Computation Offloading and Caching Solution". In: *IEEE Transactions on Intelligent Transportation Systems* 22.4 (2021), pp. 2212–2225. DOI: 10.1109/TITS.2020.2997832.

[53] Yalan Wu et al. "Load Balance Guaranteed Vehicle-to-Vehicle Computation Offloading for Min-Max Fairness in VANETs". In: *IEEE Transactions on Intelligent Transportation Systems* (2021), pp. 1–20. DOI: 10.1109/TITS.2021.3109154.

[54] Andreas Folkers, Matthias Rick, and Christof Büskens. "Controlling an Autonomous Vehicle with Deep Reinforcement Learning". In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 2025–2031. DOI: 10.1109/IVS.2019.8814124.

[55] Tong Wu et al. "Multi-Agent Deep Reinforcement Learning for Urban Traffic Light Control in Vehicular Networks". In: *IEEE Transactions on Vehicular Technology* 69.8 (2020), pp. 8243–8256. DOI: 10.1109/TVT.2020.2997896.

[56] Seyhan Ucar et al. "Vehicular Knowledge Networking and Application to Risk Reasoning". In: *Proceedings of the Twenty-First International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*. Mobihoc '20. 2020, pp. 351–356. DOI: 10.1145/3397166.3413467.

[57] Amir Rasouli, Iuliia Kotseruba, and John K. Tsotsos. "Are They Going to Cross? A Benchmark Dataset and Baseline for Pedestrian Crosswalk Behavior". In: *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. 2017, pp. 206–213. DOI: 10.1109/ICCVW.2017.33.

[58] S. Roychowdhury et al. "Machine Learning Models for Road Surface and Friction Estimation using Front-Camera Images". In: *2018 International Joint Conference on Neural Networks (IJCNN)*. 2018, pp. 1–8.

[59] Osama A. Osman et al. "Prediction of Near-Crashes from Observed Vehicle Kinematics using Machine Learning". In: *Transportation Research Record* 2673.12 (2019), pp. 463–473.

[60] Hongbo Gao et al. “Object Classification Using CNN-Based Fusion of Vision and LIDAR in Autonomous Vehicle Environment”. In: *IEEE Transactions on Industrial Informatics* 14.9 (2018), pp. 4224–4231. DOI: 10.1109/TII.2018.2822828.

[61] J. Wang et al. “A Forward Collision Warning Algorithm With Adaptation to Driver Behaviors”. In: *IEEE Transactions on Intelligent Transportation Systems* 17.4 (2016), pp. 1157–1167. DOI: 10.1109/TITS.2015.2499838.

[62] M. Ruta et al. “A Knowledge Fusion Approach for Context Awareness in Vehicular Networks”. In: *IEEE Internet of Things Journal* 5.4 (Aug. 2018), pp. 2407–2419.

[63] J. Kim and D. Kum. “Collision Risk Assessment Algorithm via Lane-Based Probabilistic Motion Prediction of Surrounding Vehicles”. In: *IEEE Transactions on Intelligent Transportation Systems* 19.9 (2018), pp. 2965–2976. DOI: 10.1109/TITS.2017.2768318.

[64] Xiaofei Ye et al. “Short-Term Prediction of Available Parking Space Based on Machine Learning Approaches”. In: *IEEE Access* 8 (2020), pp. 174530–174541. DOI: 10.1109/ACCESS.2020.3025589.

[65] Adam Thor Thorgeirsson et al. “Probabilistic Prediction of Energy Demand and Driving Range for Electric Vehicles With Federated Learning”. In: *IEEE Open Journal of Vehicular Technology* 2 (2021), pp. 151–161. DOI: 10.1109/OJVT.2021.3065529.

[66] Sophie-Grace Chappell. “Plato on Knowledge in the Theaetetus”. In: *The Stanford Encyclopedia of Philosophy*. Ed. by Edward N. Zalta. Spring 2021. Metaphysics Research Lab, Stanford University, 2021.

[67] Jonathan Jenkins Ichikawa and Matthias Steup. “The Analysis of Knowledge”. In: *The Stanford Encyclopedia of Philosophy* (2017). Ed. by Edward N. Zalta. URL: https://plato.stanford.edu/archives/sum2018/entries/knowledge-analysis/.

[68] Frieder Nake. “Data, Information, and Knowledge”. In: *Organizational Semiotics: Evolving a Science of Information Systems*. Ed. by Kecheng Liu et al. 2002, pp. 41–50. ISBN: 978-0-387-35611-2. DOI: 10.1007/978-0-387-35611-2_3.

[69] Chaim Zins. “Conceptual approaches for defining data, information, and knowledge”. In: *Journal of the American Society for Information Science and Technology* 58.4 (2007), pp. 479–493. DOI: https://doi.org/10.1002/asi.20508.

[70] Shane Legg and Marcus Hutter. “A Collection of Definitions of Intelligence”. In: *Proceedings of the 2007 Conference on Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms: Proceedings of the AGI Workshop 2006*. NLD: IOS Press, 2007, pp. 17–24. ISBN: 9781586037581.

[71] Jun Li et al. "Survey on Artificial Intelligence for Vehicles". In: *Automotive Innovation* 1.1 (Jan. 2018), pp. 2–14. ISSN: 2522-8765. DOI: 10.1007/s42154-018-0009-9.

[72] Wang Tong et al. "Artificial Intelligence for Vehicle-to-Everything: A Survey". In: *IEEE Access* 7 (2019), pp. 10823–10843. DOI: 10.1109/ACCESS.2019.2891073.

[73] Yifang Ma et al. "Artificial intelligence applications in the development of autonomous vehicles: a survey". In: *IEEE/CAA Journal of Automatica Sinica* 7.2 (2020), pp. 315–329. DOI: 10.1109/JAS.2020.1003021.

[74] Hao Ye et al. "Machine Learning for Vehicular Networks: Recent Advances and Application Examples". In: *IEEE Vehicular Technology Magazine* 13.2 (2018), pp. 94–101. DOI: 10.1109/MVT.2018.2811185.

[75] Fengxiao Tang et al. "Comprehensive Survey on Machine Learning in Vehicular Network: Technology, Applications and Challenges". In: *IEEE Communications Surveys & Tutorials* (2021), pp. 1–1. DOI: 10.1109/COMST.2021.3089688.

[76] Xiaoyuan Liang et al. "A Deep Reinforcement Learning Network for Traffic Light Cycle Control". In: *IEEE Transactions on Vehicular Technology* 68.2 (2019), pp. 1243–1253. DOI: 10.1109/TVT.2018.2890726.

[77] Shiliang Sun et al. "A Survey of Optimization Methods From a Machine Learning Perspective". In: *IEEE Transactions on Cybernetics* 50.8 (2020), pp. 3668–3681. DOI: 10.1109/TCYB.2019.2950779.

[78] V. Praveen et al. "A Survey on Various Optimization Algorithms to Solve Vehicle Routing Problem". In: *2019 5th International Conference on Advanced Computing Communication Systems (ICACCS)*. 2019, pp. 134–137. DOI: 10.1109/ICACCS.2019.8728417.

[79] Bilal et al. "Differential Evolution: A review of more than two decades of research". In: *Engineering Applications of Artificial Intelligence* 90 (2020), p. 103479. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2020.103479.

[80] Michalis Mavrovouniotis, Changhe Li, and Shengxiang Yang. "A survey of swarm intelligence for dynamic optimization: Algorithms and applications". In: *Swarm and Evolutionary Computation* 33 (2017), pp. 1–17. ISSN: 2210-6502.

[81] Maryam Karimi-Mamaghan et al. "Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art". In: *European Journal of Operational Research* (2021). ISSN: 0377-2217. DOI: https://doi.org/10.1016/j.ejor.2021.04.032.

[82] Madjid Tavana and Vahid Hajipour. "A practical review and taxonomy of fuzzy expert systems: methods and applications". In: *Benchmarking: An International Journal* 27.1 (Jan. 2020), pp. 81–136. ISSN: 1463-5771. DOI: 10.1108/BIJ-04-2019-0178.

[83] K.V. Shihabudheen and G.N. Pillai. "Recent advances in neuro-fuzzy system: A survey". In: *Knowledge-Based Systems* 152 (2018), pp. 136–162. ISSN: 0950-7051. DOI: 10.1016/j.knosys.2018.04.014.

[84] Shaoxiong Ji et al. "A Survey on Knowledge Graphs: Representation, Acquisition, and Applications". In: *IEEE Transactions on Neural Networks and Learning Systems* (2021), pp. 1–21. DOI: 10.1109/TNNLS.2021.3070843.

[85] Dan Brickley and R.V. Guha. *RDF Schema 1.1*. First Edition of a Recommendation. W3C, Feb. 2014. URL: https://www.w3.org/TR/2014/REC-rdf-schema-20140225.

[86] Benjamin Klotz et al. "VSSo - A vehicle signal and attribute ontology". In: *SSN 2018, 9th International Semantic Sensor Networks Workshop*. Monterey, USA, Oct. 2018.

[87] W3C OWL Working Group. *OWL 2 Web Ontology Language Document Overview (Second Edition)*. Second Edition of a Recommendation. W3C, Dec. 2012. URL: https://www.w3.org/TR/2012/REC-owl2-overview-20121211.

[88] Steve Harris, Andy Seaborne, and Eric Prud'hommeaux. *SPARQL 1.1 Query Language*. First Edition of a Recommendation. W3C, Mar. 2013. URL: https://www.w3.org/TR/2013/REC-sparql11-query-20130321.

[89] Birte Glimm et al. *SPARQL 1.1 Entailment Regimes*. First Edition of a Recommendation. W3C, Mar. 2013. URL: https://www.w3.org/TR/2013/REC-sparql11-entailment-20130321.

[90] David Martin et al. *OWL-S: Semantic Markup for Web Services*. W3C Member Submission. W3C, Nov. 2004. URL: https://www.w3.org/Submission/2004/SUBM-OWL-S-20041122.

[91] Sebastian Kaebisch et al. *Web of Things (WoT) Thing Description*. First Edition of a Recommendation. W3C, Apr. 2020. URL: https://www.w3.org/TR/2020/REC-wot-thing-description-20200409.

[92] Andrei Ciortea, Olivier Boissier, and Alessandro Ricci. "Engineering World-Wide Multi-Agent Systems with Hypermedia". In: *Engineering Multi-Agent Systems*. Ed. by Danny Weyns, Viviana Mascardi, and Alessandro Ricci. Cham: Springer International Publishing, 2019, pp. 285–301. ISBN: 978-3-030-25693-7.

[93] Automotive Edge Computing Consortium (AECC). *Driving Data to the Edge: The Challenge of Traffic Distribution*. https://aecc.org/driving-data-to-deliver-addressing-the-connected-vehicle-data-challenge. 2020.

[94] Wenjie Liu et al. "Mobility-Aware Coded Edge Caching in Vehicular Networks with Dynamic Content Popularity". In: *2021 IEEE Wireless Communications and Networking Conference (WCNC)*. 2021, pp. 1–6. DOI: 10.1109/WCNC49053.2021.9417383.

[95] Zhou Su et al. "Contract Based Edge Caching in Vehicular Networks". In: *The Next Generation Vehicular Networks, Modeling, Algorithm and Applications*. Cham: Springer International Publishing, 2021, pp. 49–67. ISBN: 978-3-030-56827-6. DOI: 10.1007/978-3-030-56827-6_3.

[96] *ETSI EN 302 895 V1.1.1 : Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Local Dynamic Map (LDM)*. Tech. rep. ETSI, Sept. 2014. URL: https://www.etsi.org/deliver/etsi_en/302800_302899/302895/01.01.01_60/en_302895v010101p.pdf.

[97] *ETSI TR 102 863 V1.1.1 : Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Local Dynamic Map (LDM); Rationale for and guidance on standardization*. Tech. rep. ETSI, June 2011. URL: https://www.etsi.org/deliver/etsi_tr/102800_102899/102863/01.01.01_60/tr_102863v010101p.pdf.

[98] *ETSI TS 102 894-2 V1.3.1 : Intelligent Transport Systems (ITS); Users and applications requirements; Part 2: Applications and facilities layer common data dictionary*. Tech. rep. ETSI, Aug. 2018. URL: https://www.etsi.org/deliver/etsi_ts/102800_102899/10289402/01.03.01_60/ts_10289402v010301p.pdf.

[99] Ming-Kai Jiau et al. "Multimedia Services in Cloud-Based Vehicular Networks". In: *IEEE Intelligent Transportation Systems Magazine* 7.3 (2015), pp. 62–79. DOI: 10.1109/MITS.2015.2417974.

[100] Aida Ghazizadeh and Stephan Olariu. "Vehicular Clouds: A Survey and Future Directions". In: *Cloud Computing for Optimization: Foundations, Applications, and Challenges*. Ed. by Bhabani Shankar Prasad Mishra et al. Cham: Springer International Publishing, 2018, pp. 435–463. ISBN: 978-3-319-73676-1. DOI: 10.1007/978-3-319-73676-1_17.

[101] Stephan Olariu. "A Survey of Vehicular Cloud Research: Trends, Applications and Challenges". In: *IEEE Transactions on Intelligent Transportation Systems* 21.6 (2020), pp. 2648–2663. DOI: 10.1109/TITS.2019.2959743.

[102] Takamasa Higuchi et al. "On the feasibility of vehicular micro clouds". In: *2017 IEEE Vehicular Networking Conference (VNC)*. 2017, pp. 179–182. DOI: 10.1109/VNC.2017.8275621.

[103] Rudzidatul Akmam Dziyauddin et al. "Computation offloading and content caching and delivery in Vehicular Edge Network: A survey". In: *Computer Networks* 197 (2021), p. 108228. ISSN: 1389-1286. DOI: 10.1016/j.comnet.2021.108228.

[104] Lei Liu et al. "Vehicular Edge Computing and Networking: A Survey". In: *Mobile Networks and Applications* (July 2020). ISSN: 1572-8153. DOI: 10.1007/s11036-020-01624-1.

[105] Le Thanh Tan, Rose Qingyang Hu, and Lajos Hanzo. "Twin-Timescale Artificial Intelligence Aided Mobility-Aware Edge Caching and Computing in Vehicular Networks". In: *IEEE Transactions on Vehicular Technology* 68.4 (2019), pp. 3086–3099. DOI: 10.1109/TVT.2019.2893898.

[106] Ke Zhang et al. "Digital Twin Empowered Content Caching in Social-Aware Vehicular Edge Networks". In: *IEEE Transactions on Computational Social Systems* (2021), pp. 1–13. DOI: 10.1109/TCSS.2021.3068369.

[107] Huaqing Wu et al. "Delay-Minimized Edge Caching in Heterogeneous Vehicular Networks: A Matching-Based Approach". In: *IEEE Transactions on Wireless Communications* 19.10 (2020), pp. 6409–6424. DOI: 10.1109/TWC.2020.3003339.

[108] Yao Zhang et al. "Towards Hit-Interruption Tradeoff in Vehicular Edge Caching: Algorithm and Analysis". In: *IEEE Transactions on Intelligent Transportation Systems* (2021), pp. 1–13. DOI: 10.1109/TITS.2021.3052355.

[109] Xiaolong Xu et al. "Edge Content Caching with Deep Spatiotemporal Residual Network for IoV in Smart City". In: *ACM Trans. Sen. Netw.* 17.3 (June 2021). ISSN: 1550-4859. DOI: 10.1145/3447032.

[110] Yuping Xing et al. "Deep Reinforcement Learning for Cooperative Edge Caching in Vehicular Networks". In: *2021 13th International Conference on Communication Software and Networks (ICCSN)*. 2021, pp. 144–149. DOI: 10.1109/ICCSN52437.2021.9463666.

[111] Jiayin Chen et al. "Cooperative Edge Caching With Location-Based and Popular Contents for Vehicular Networks". In: *IEEE Transactions on Vehicular Technology* 69.9 (2020), pp. 10291–10305. DOI: 10.1109/TVT.2020.3004720.

[112] Anitha Veerasamy et al. "Angle and Context Free Grammar Based Precarious Node Detection and Secure Data Transmission in MANETs". In: *The Scientific World Journal* 2016 (Jan. 2016). ISSN: 2356-6140. DOI: 10.1155/2016/3596345.

[113] S. Corson and J. Macker. *Mobile Ad hoc Networking (MANET): Routing Protocol Performance Issues and Evaluation Considerations*. RFC 2501. RFC Editor, Jan. 1999. URL: https://www.rfc-editor.org/rfc/rfc2501.txt.

[114] Shafinaz Buruhanudeen et al. "Existing MANET routing protocols and metrics used towards the efficiency and reliability- an overview". In: *2007 IEEE International Conference on Telecommunications and Malaysia International Conference on Communications*. 2007, pp. 231–236. DOI: 10.1109/ICTMICC.2007.4448632.

[115] T. Clausen and P. Jacquet. *Optimized Link State Routing Protocol (OLSR)*. RFC 3626. RFC Editor, Oct. 2003. URL: https://www.rfc-editor.org/rfc/rfc3626.txt.

[116] E. Belding-Royer C. Perkins and S. Das. *Ad hoc On-Demand Distance Vector (AODV) Routing*. RFC 3561. RFC Editor, July 2003. URL: https://www.rfc-editor.org/rfc/rfc3561.txt.

[117] Y. Hu D. Johnson and D. Maltz. *The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4*. RFC 4728. RFC Editor, Feb. 2007. URL: https://www.rfc-editor.org/rfc/rfc4728.txt.

[118] Z.J. Haas. "A new routing protocol for the reconfigurable wireless networks". In: *Proceedings of ICUPC 97 - 6th International Conference on Universal Personal Communications*. Vol. 2. 1997, 562–566 vol.2. DOI: 10.1109/ICUPC.1997.627227.

[119] Boubakr Nour et al. "A survey of Internet of Things communication using ICN: A use case perspective". In: *Computer Communications* 142-143 (2019), pp. 95–123. ISSN: 0140-3664. DOI: 10.1016/j.comcom.2019.05.010.

[120] O. Serhane et al. "A Survey of ICN Content Naming and In-Network Caching in 5G and Beyond Networks". In: *IEEE Internet of Things Journal* 8.6 (2021), pp. 4081–4104. DOI: 10.1109/JIOT.2020.3022243.

[121] Muhammad Atif Ur Rehman, Rehmat Ullah, and Byung Seo Kim. "NINQ: Name-Integrated Query Framework for Named-Data Networking of Things". In: *Sensors* 19.13 (2019). ISSN: 1424-8220. DOI: 10.3390/s19132906.

[122] Lucas Wang et al. "Data naming in Vehicle-to-Vehicle communications". In: *2012 Proceedings IEEE INFOCOM Workshops*. 2012, pp. 328–333. DOI: 10.1109/INFCOMW.2012.6193515.

[123] Michał Król and Ioannis Psaras. "NFaaS: Named Function as a Service". In: *Proceedings of the 4th ACM Conference on Information-Centric Networking*. ICN '17. Berlin, Germany: Association for Computing Machinery, 2017, pp. 134–144. ISBN: 9781450351225. DOI: 10.1145/3125719.3125727.

[124] Y. Kumamoto, H. Yoshii, and H. Nakazato. "Real-World Implementation of Function Chaining in Named Data Networking for IoT Environments". In: *2020 IEEE International Workshop Technical Committee on Communications Quality and Reliability (CQR)*. 2020, pp. 1–6. DOI: 10.1109/CQR47547.2020.9101396.

[125] D. Wu et al. "Vision and Challenges for Knowledge Centric Networking". In: *IEEE Wireless Communications* 26.4 (Aug. 2019), pp. 117–123.

[126] H. Hao et al. "Knowledge-centric proactive edge caching over mobile content distribution network". In: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2018, pp. 450–455.

[127] X. Zhang, H. Wang, and H. Zhao. "An SDN framework for UAV backbone network towards knowledge centric networking". In: *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2018, pp. 456–461.

[128] D. Sapra and A. D. Pimentel. "Deep Learning Model Reuse and Composition in Knowledge Centric Networking". In: *2020 29th International Conference on Computer Communications and Networks (ICCCN)*. 2020, pp. 1–11.

[129] Alejandro Ivan Morales Medina, Nathan van de Wouw, and Henk Nijmeijer. "Cooperative Intersection Control Based on Virtual Platooning". In: *IEEE Transactions on Intelligent Transportation Systems* 19.6 (2018), pp. 1727–1740. DOI: 10.1109/TITS.2017.2735628.

[130] Karl Kurzer, Florian Engelhorn, and J. Marius Zöllner. "Decentralized Cooperative Planning for Automated Vehicles with Continuous Monte Carlo Tree Search". In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 452–459. DOI: 10.1109/ITSC.2018.8569988.

[131] Peng Liu, Arda Kurt, and Umit Ozguner. "Distributed Model Predictive Control for Cooperative and Flexible Vehicle Platooning". In: *IEEE Transactions on Control Systems Technology* 27.3 (2019), pp. 1115–1128. DOI: 10.1109/TCST.2018.2808911.

[132] Yara Rizk, Mariette Awad, and Edward W. Tunstel. "Decision Making in Multiagent Systems: A Survey". In: *IEEE Transactions on Cognitive and Developmental Systems* 10.3 (2018), pp. 514–529. DOI: 10.1109/TCDS.2018.2840971.

[133] Dave Conway-Jones et al. "Demonstration of Federated Learning in a Resource-Constrained Networked Environment". In: *2019 IEEE International Conference on Smart Computing (SMARTCOMP)*. 2019, pp. 484–486. DOI: 10.1109/SMARTCOMP.2019.00095.

[134] Joost Verbraeken et al. "A Survey on Distributed Machine Learning". In: *ACM Comput. Surv.* 53.2 (Mar. 2020). ISSN: 0360-0300. DOI: 10.1145/3377454.

[135] Peter Kairouz et al. "Advances and Open Problems in Federated Learning". In: *Foundations and Trends® in Machine Learning* 14.1 (2021). ISSN: 1935-8237. DOI: 10.1561/2200000083.

[136] Thanh Thi Nguyen, Ngoc Duy Nguyen, and Saeid Nahavandi. "Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications". In: *IEEE Transactions on Cybernetics* 50.9 (2020), pp. 3826–3839. DOI: 10.1109/TCYB.2020.2977374.

[137] Chi Zhang et al. "Distributed Multi-Task Classification: A Decentralized Online Learning Approach". In: *Machine Learning* 107.4 (Apr. 2018), pp. 727–747. ISSN: 0885-6125. DOI: 10.1007/s10994-017-5676-y.

[138] Michael Kamp et al. "Efficient Decentralized Deep Learning by Dynamic Model Averaging". In: *Machine Learning and Knowledge Discovery in Databases*. Ed. by Michele Berlingerio et al. Cham: Springer International Publishing, 2019, pp. 393–409. ISBN: 978-3-030-10925-7.

[139] Behrouz Pourghebleh, Vahideh Hayyolalam, and Amir Aghaei Anvigh. "Service discovery in the Internet of Things: review of current trends and research challenges". In: *Wireless Networks* 26.7 (Oct. 2020), pp. 5371–5391. ISSN: 1572-8196. DOI: 10.1007/s11276-020-02405-0.

[140] Takayuki Nishio and Ryo Yonetani. "Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge". In: *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)* (May 2019).

[141] Luping Wang, Wei Wang, and Bo Li. "CMFL: Mitigating Communication Overhead for Federated Learning". In: *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)* (2019), pp. 954–964.

[142] Richeng Jin, Xiaofan He, and Huaiyu Dai. *On the Design of Communication Efficient Federated Learning over Wireless Networks*. 2020. arXiv: 2004.07351 [cs.LG].

[143] Jinjin Xu et al. *Ternary Compression for Communication-Efficient Federated Learning*. 2020. arXiv: 2003.03564 [cs.LG].

[144] Joohyung Jeon et al. "Optimal User Selection for High-Performance and Stabilized Energy-Efficient Federated Learning Platforms". In: *Electronics* 9.9 (2020). ISSN: 2079-9292. DOI: 10.3390/electronics9091359.

[145] Bo Xu et al. "Dynamic Client Association for Energy-Aware Hierarchical Federated Learning". In: *2021 IEEE Wireless Communications and Networking Conference (WCNC)*. 2021, pp. 1–6. DOI: 10.1109/WCNC49053.2021.9417267.

[146] Junjie Pang et al. "Realizing the Heterogeneity: A Self-Organized Federated Learning Framework for IoT". In: *IEEE Internet of Things Journal* 8.5 (2021), pp. 3088–3098. DOI: 10.1109/JIOT.2020.3007662.

[147] Hao Wang et al. "Optimizing Federated Learning on Non-IID Data with Reinforcement Learning". In: *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. 2020, pp. 1698–1707. DOI: 10.1109/INFOCOM41043.2020.9155494.

[148] Christopher Briggs, Zhong Fan, and Peter Andras. "Federated learning with hierarchical clustering of local updates to improve training on non-IID data". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. 2020, pp. 1–9. DOI: 10.1109/IJCNN48605.2020.9207469.

[149] Xin Yao et al. "Towards Faster and Better Federated Learning: A Feature Fusion Approach". In: *2019 IEEE International Conference on Image Processing (ICIP)*. 2019, pp. 175–179. DOI: 10.1109/ICIP.2019.8803001.

[150] Rong Yu and Peichun Li. "Toward Resource-Efficient Federated Learning in Mobile Edge Computing". In: *IEEE Network* 35.1 (2021), pp. 148–155. DOI: 10.1109/MNET.011.2000295.

[151] Yue Zhao et al. *Federated Learning with Non-IID Data*. 2018. arXiv: 1806.00582 [cs.LG].

[152] Eunjeong Jeong et al. "Multi-hop Federated Private Data Augmentation with Sample Compression". In: *IJCAI Workshop on Federated Machine Learning for User Privacy and Data Confidentiality (FML'19)*. Macau, Aug. 2019.

[153] Eugene Bagdasaryan et al. *How To Backdoor Federated Learning*. 2018. arXiv: 1807.00459 [cs.CR].

[154] Z. Li, V. Sharma, and S. P. Mohanty. "Preserving Data Privacy via Federated Learning: Challenges and Solutions". In: *IEEE Consumer Electronics Magazine* 9.3 (2020), pp. 8–16.

[155] Yuwei Wang and Burak Kantarci. "A Novel Reputation-aware Client Selection Scheme for Federated Learning within Mobile Environments". In: *2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. 2020, pp. 1–6. DOI: 10.1109/CAMAD50429.2020.9209263.

[156] Yuan Liu et al. "FedCoin: A Peer-to-Peer Payment System for Federated Learning". In: *Federated Learning: Privacy and Incentive*. Ed. by Qiang Yang, Lixin Fan, and Han Yu. Cham: Springer International Publishing, 2020, pp. 125–138. ISBN: 978-3-030-63076-8. DOI: 10.1007/978-3-030-63076-8_9.

[157] Han Yu et al. "A Fairness-Aware Incentive Scheme for Federated Learning". In: *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. AIES '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 393–399. ISBN: 9781450371100.

[158] B. Swenson et al. "Distributed Gradient Descent: Nonconvergence to Saddle Points and the Stable-Manifold Theorem". In: *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*. 2019, pp. 595–601.

[159] S. Wang et al. "Adaptive Federated Learning in Resource Constrained Edge Computing Systems". In: *IEEE Journal on Selected Areas in Communications* 37.6 (2019), pp. 1205–1221.

[160] Claudia Campolo et al. "Towards Named AI Networking: Unveiling the Potential of NDN for Edge AI". In: *Ad-Hoc, Mobile, and Wireless Networks*. Ed. by Luigi Alfredo Grieco et al. 2020, pp. 16–22. ISBN: 978-3-030-61746-2.

[161] Michael R. Genesereth. *KIF. Knowledge Interchange Format: Draft proposed American National Standard (dpANS). Technical Report 2/98-004*. Tech. rep. 1998. URL: http://logic.stanford.edu/kif/dpans.html.

[162] Vitor Silva, Marite Kirikova, and Gundars Alksnis. "Containers for Virtualization: An Overview". In: *Applied Computer Systems* 23 (May 2018), pp. 21–27. DOI: 10.2478/acss-2018-0003.

[163] N. Lal, S. Kumar, and V. K. Chaurasiya. "An efficient update strategy for content synchronization in Content-Centric Networking (CCN)". In: *China Communications* 16.1 (2019), pp. 108–118.

[164] W. Navidi and T. Camp. "Stationary distributions for the random waypoint mobility model". In: *IEEE Transactions on Mobile Computing* 3.1 (2004), pp. 99–108. DOI: 10.1109/TMC.2004.1261820.

[165] Ahmed Imteaj et al. *Federated Learning for Resource-Constrained IoT Devices: Panoramas and State-of-the-art*. 2020. arXiv: 2002.10610 [cs.LG].

[166] Wei Yang Bryan Lim et al. *Federated Learning in Mobile Edge Networks: A Comprehensive Survey*. 2019. arXiv: 1909.11875 [cs.NI].

[167] Ahmet M. Elbir, Burak Soner, and Sinem Coleri. *Federated Learning in Vehicular Networks*. 2020. arXiv: 2006.01412 [eess.SP].

[168] Sebastian Caldas et al. *LEAF: A Benchmark for Federated Settings*. 2019. arXiv: 1812.01097 [cs.LG].

[169] C. Wu, A. M. Bayen, and A. Mehta. "Stabilizing Traffic with Autonomous Vehicles". In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. May 2018, pp. 6012–6018.

[170] L. Alfaseeh, S. Djavadian, and B. Farooq. "Impact of Distributed Routing of Intelligent Vehicles on Urban Traffic". In: *2018 IEEE International Smart Cities Conference (ISC2)*. Sept. 2018, pp. 1–7.

[171] Stephen M. Casner, Edwin L. Hutchins, and Don Norman. "The Challenges of Partially Automated Driving". In: *Commun. ACM* 59.5 (Apr. 2016), pp. 70–77. ISSN: 0001-0782.

[172] M. Yu, R. Vasudevan, and M. Johnson-Roberson. "Occlusion-Aware Risk Assessment for Autonomous Driving in Urban Environments". In: *IEEE Robotics and Automation Letters* 4.2 (Apr. 2019), pp. 2235–2241. ISSN: 2377-3774.

[173] On-Road Automated Driving (ORAD) committee. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles*. 2018. DOI: 10.4271/J3016_201806.

[174] S. G. McGill et al. "Probabilistic Risk Metrics for Navigating Occluded Intersections". In: *IEEE Robotics and Automation Letters* 4.4 (Oct. 2019), pp. 4322–4329.

[175] S. Zang et al. "The Impact of Adverse Weather Conditions on Autonomous Vehicles: How Rain, Snow, Fog, and Hail Affect the Performance of a Self-Driving Car". In: *IEEE Vehicular Technology Magazine* 14.2 (June 2019), pp. 103–111.

[176] Oliver Pink, Jan Becker, and Sören Kammel. "Automated driving on public roads: Experiences in real traffic". In: *it - Information Technology* 57 (July 2015), pp. 223–230.

[177] Xin Xin et al. "A Literature Review of the Research on Take-Over Situation in Autonomous Driving". In: *International Conference on Human-Computer Interaction*. July 2019, pp. 160–169.

[178] A. Zyner, S. Worrall, and E. Nebot. "A Recurrent Neural Network Solution for Predicting Driver Intention at Unsignalized Intersections". In: *IEEE Robotics and Automation Letters* 3.3 (July 2018), pp. 1759–1764.

[179] J. F. Fisac et al. "Hierarchical Game-Theoretic Planning for Autonomous Vehicles". In: *2019 International Conference on Robotics and Automation (ICRA)*. May 2019, pp. 9590–9596.

[180] Andreas Reschka and Markus Maurer. "Conditions for a safe state of automated road vehicles". In: *it - Information Technology* 57 (2015), pp. 215–222.

[181] A. Rasouli and J. K. Tsotsos. "Autonomous Vehicles That Interact With Pedestrians: A Survey of Theory and Practice". In: *IEEE Transactions on Intelligent Transportation Systems* (2019), pp. 1–19.

[182] Daniel Krajzewicz et al. "Recent Development and Applications of SUMO - Simulation of Urban MObility". In: *International Journal On Advances in Systems and Measurements* 5.3&4 (Dec. 2012), pp. 128–138.

[183] Lara Codeca and Jérôme Härri. "Monaco SUMO Traffic (MoST) Scenario: A 3D Mobility Scenario for Cooperative ITS". In: *SUMO User Conference*. B, May 2018.

[184] Stefan Hausberger and Daniel Krajzewicz. *COLOMBO Deliverable 4.2: Extended Simulation Tool PHEM coupled to SUMO with User Guide*. Tech. rep. Feb. 2014.

[185] Mark Morando et al. "Studying the Safety Impact of Autonomous Vehicles Using Simulation-Based Surrogate Safety Measures". In: *Journal of advanced transportation* (2018). DOI: 10.1155/2018/6135183.

[186] Aleksandra Deluka Tibljaš et al. "Introduction of Autonomous Vehicles: Roundabouts Design and Safety Performance Evaluation". In: *Sustainability* 10.4 (2018). DOI: 10.3390/su10041060.

[187] Navreet Virdi et al. "A safety assessment of mixed fleets with Connected and Autonomous Vehicles using the Surrogate Safety Assessment Module". In: *Accident Analysis & Prevention* 131 (2019), pp. 95–111. DOI: 10.1016/j.aap.2019.06.001.

[188] Jade Montgomery, Kristofer D. Kusano, and Hampton C. Gabler. "Age and Gender Differences in Time to Collision at Braking From the 100-Car Naturalistic Driving Study". In: *Traffic Injury Prevention* 15.sup1 (2014), S15–S20. DOI: 10.1080/15389588.2014.928703.

[189] Richard Van Der Horst and Jeroen Hogema. "Time-to-collision and collision avoidance systems". In: 6th ICTCT workshop. Salzburg, Austria, Jan. 1994.

[190] Kristofer Kusano and Hampton Gabler. "Method for Estimating Time to Collision at Braking in Real-World, Lead Vehicle Stopped Rear-End Crashes for Use in Pre-Crash System Design". In: *SAE Int J Passeng Cars - Mech Syst* 4 (June 2011), pp. 435–443. DOI: 10.4271/2011-01-0576.

[191] R. Krajewski et al. "The rounD Dataset: A Drone Dataset of Road User Trajectories at Roundabouts in Germany". In: *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 2020, pp. 1–6. DOI: 10.1109/ITSC45102.2020.9294728.

[192] Assaf Botzer and Oren Musicant. "Driver Choices of Time-to-Collision Thresholds for a Collision Warning System". In: 2016. DOI: 10.1145/2970930.2970958.

[193] Michiel M. Minderhoud and Piet H.L. Bovy. "Extended time-to-collision measures for road traffic safety assessment". In: *Accident Analysis & Prevention* 33.1 (2001), pp. 89–97. DOI: 10.1016/S0001-4575(00)00019-1.

[194] Chuang Qian et al. "Cooperative GNSS-RTK Ambiguity Resolution with GNSS, INS, and LiDAR Data for Connected Vehicles". In: *Remote Sensing* 12.6 (2020). DOI: 10.3390/rs12060949.

[195] C. E. Shannon. "A Mathematical Theory of Communication". In: *Bell System Technical Journal* 27.3 (1948), pp. 379–423. DOI: 10.1002/j.1538-7305.1948.tb01338.x.

[196] Wei Zhan et al. "INTERACTION Dataset: An INTERnational, Adversarial and Cooperative moTION Dataset in Interactive Driving Scenarios with Semantic Maps". In: *arXiv:1910.03088 [cs, eess]* (Sept. 2019).

[197] M. Aksela. "Comparison of Classifier Selection Methods for Improving Committee Performance". In: *Multiple Classifier Systems*. 2003.

[198] Julien Meynet and Jean-Philippe Thiran. "Information theoretic combination of pattern classifiers". In: *Pattern Recognition* 43.10 (2010), pp. 3412–3421. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2010.04.013.

[199] Morteza Noshad, Yu Zeng, and Alfred O. Hero. "Scalable Mutual Information Estimation Using Dependence Graphs". In: *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2019, pp. 2962–2966. DOI: 10.1109/ICASSP.2019.8683351.

[200] Frank Nielsen. "Hierarchical Clustering". In: *Introduction to HPC with MPI for Data Science*. Cham: Springer International Publishing, 2016, pp. 195–211. ISBN: 978-3-319-21903-5. DOI: 10.1007/978-3-319-21903-5_8.

[201] Kamran Khan et al. "DBSCAN: Past, present and future". In: *The Fifth International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2014)*. 2014, pp. 232–238. DOI: 10.1109/ICADIWT.2014.6814687.

[202] J. Chang, Cheol Oh, and Mingway Chang. "Effects of traffic condition (v/c) on safety at freeway facility sections". In: *Transportation Research Circular* E-C018 (2000), pp. 200–208. ISSN: 0097-8515.

[203] Ning Wu. "A universal procedure for capacity determination at unsignalized (priority-controlled) intersections". In: *Transportation Research Part B: Methodological* 35.6 (2001), pp. 593–623. ISSN: 0191-2615. DOI: 10.1016/S0191-2615(00)00012-6.

[204] Ramu Arroju et al. "Comparative evaluation of roundabout capacities under heterogeneous traffic conditions". In: *Journal of Modern Transportation* 23.4 (Dec. 2015), pp. 310–324. ISSN: 2196-0577. DOI: 10.1007/s40534-015-0089-8.

[205] Rahmi Akçelik. "An Assessment of the Highway Capacity Manual Edition 6 Roundabout Capacity Model". In: *5th International Roundabout Conference, Transportation Research Board*. May 2017.

[206] W. Brilon, N. Wu, and L. Bondzio. "Unsignalized Intersections in Germany - a State of the Art 1997". In: *Third International Symposium on Intersections without Traffic Signals*. 1997.

[207] João Mendes-Moreira et al. "Ensemble Approaches for Regression: A Survey". In: *ACM Comput. Surv.* 45.1 (Dec. 2012). ISSN: 0360-0300. DOI: 10.1145/2379776.2379786.

[208] Seyhan Uçar et al. "Vehicular Knowledge Networking and Mobility-Aware Smart Knowledge Placement". In: *IEEE Consumer Communications & Networking Conference (CNCC) 2022*. To appear. 2022.

[209] T. Henderson. *Network Simulations with the ns-3 Simulator*. 2008.

[210] Spyridon Mastorakis, Alexander Afanasyev, and Lixia Zhang. "On the Evolution of ndnSIM: an Open-Source Simulator for NDN Experimentation". In: *ACM Computer Communication Review* (July 2017).

[211] T. Henderson. *The ns Manual: Simulator Basics; Mobile Networking in ns; The basic wireless model in ns; Creating Node movements*. Tech. rep. Nov. 2011. URL: https://www.isi.edu/nsnam/ns/doc/node172.html.

[212] "IEEE Standard for Wireless Access in Vehicular Environments (WAVE) – Multi-Channel Operation". In: *IEEE Std 1609.4-2016 (Revision of IEEE Std 1609.4-2010)* (2016), pp. 1–94. DOI: 10.1109/IEEESTD.2016.7435228.