Building a Lane Merge Coordination for Connected Vehicles Using Deep Reinforcement Learning

Omar Nassef, Luis Sequeira, Elias Salam, and Toktam Mahmoodi

***Abstract*—This paper presents a data-driven framework for trajectory recommendation in automated and cooperative driving. The considered cooperative driving manoeuvre is lane-merge coordination, and while the trajectory recommendation can only be communicated to the connected vehicles, in computation of those recommendations both connected and unconnected vehicles are taken into account. The data-driven framework is implemented centrally, comprising of two main components of a *Traffic Orchestrator* and *Data Fusion*. The *Traffic Orchestrator* predicts the safest trajectories for connected vehicles involved in the lane-merge manoeuvre. The *Data Fusion* incorporates camera detected vehicles in order to map all vehicles including connected and unconnected. To this end, the recommendations are built using various state-of-the-art machine learning techniques including deep reinforcement learning and dueling deep Q-network. Our evaluations are conducted using the real-system deployed in the test track, with a mix of connected and unconnected vehicles. The results demonstrate precision of predicted trajectories, and percentage of successful lane merge achieved deploying different machine learning techniques.**

***Index Terms*—Lane merge, intelligent transport system, V2X communications, reinforcement learning, machine learning.**

I. Introduction

Intelligent Transport System (ITS) enables the generation of extensive and detailed data relating to vehicles in addition to the environment of their operation. This data can be used to improve road safety and provide better driving experience. Connected vehicles are capable of transmitting and receiving information in order to contribute to this data pool, but also to fully benefit from exploitation of such data. Additionally connected vehicles together with the road infrastructure shape the Internet of Things (IoT) network to enable the intelligent transportation infrastructure [1], [2]. Associations such as the European Telecommunications Standards Institute (ETSI) and $5G$ Automotive Association (5GAA) have promoted the use of cellular Vehicle-to-Everything (V2X) communications in order to enhance road safety, traffic efficiency, reduce environmental issues and energy costs [3]. Due to the advancements in 5^{th} Generation Mobile Network ($5G$) and *V2X*, many use cases and applications are under research and development for cooperative driving such as cooperative collision avoidance, high density platooning and cooperative lane merge [4] [5].

O. Nassef, L. Sequeira and T. Mahmoodi are with the Department of Engineering at King's College London. Email: {omar.nassef, luis.sequeira, toktam.mahmoodi}@kcl.ac.uk.

E. Salam is with the Department of Software at Orange Labs Services. Email: elias.salam@orange.com.

In this paper, we focus on cooperative lane merge scenario where a vehicle is merging into a carriageway between two vehicles. The proposed coordination model is based on a centralised system which is analysed and evaluated in detail. *Reinforcement Learning (RL)* is used by the system in order to deliver trajectory recommendations to connected vehicles, accounting for all surrounding vehicles (i.e., connected and unconnected). Time-critical variables such as location, speed and acceleration are used as input to the deep reinforcement learning model. Furthermore, two different *RL* algorithms are presented and evaluated to ascertain whether a merging vehicle can perform a manoeuvre safely. Additionally, results from the optimal model tested on real connected vehicles on the test track [6] are described. The contributions of this paper are as follow:

- A *Traffic Orchestrator (TO)* model based on a centralised system, that delivers trajectory recommendations to connected vehicles.
- Performance evaluation of two different RL approaches: Dueling Deep Q-Network (Dueling DQN) and Deep Q-Network (DQN). The models explore different reward functions, utilising Newtonian Law of Motion equations for trajectory recommendations.
- A comparison between the different RL models with state-of-the-art Machine Learning (ML) models for lane merging.
- Contrast between best performing RL model with human-recorded trajectories, with respect to acceleration, inter-vehicle safety distance and way-point position
- A *Data Fusion (DF)* model that synergies the centralised micro service oriented architecture, delivering descriptors of connected and unconnected vehicles to the *TO*.

Two different algorithms presented in this work are thoroughly explored: DQN and Dueling DQN. The Dueling DQN showed the most optimal results, providing human-like trajectories with very low bias. The inter vehicle distance, acceleration, individual positions and manoeuvre distance in trajectory recommendation are evaluated extensively to deduce the performance of incorporating such model.

This paper is organised as follow: Section II shows the State-of-the-Art gauging the different *RL* approaches and other methods for lane merge algorithms. Section III provides the general system model, while section IV describes the decisions taken in the formulation of the *RL* model. Section V outlines the necessity and integration of the microservices on the proposed architecture. The analysis of the the deep *RL* algorithms is explored in section VI on the data-set it was trained on.

This section also describes the real world scenario and tests with the accompanying obtained Key Performance Indicators (KPIs). Finally, we draw the conclusion and highlight some future works in section VII.

II. State of the Art

A. Lane merge coordination

Vehicles have gained intelligence as self-driving capabilities have increased significantly in the previous years. Cruise Control, Lane Following and assisted large lateral control manoeuvres, such as lane changes appeared paving a way for progression towards topics like Cooperative Adaptive Cruise Control (CACC) for lane change and merge [7]. In this context, there are several situations that impact vehicle behaviour, such as weather, road grade or surrounding vehicles. In this work we will focus on a lane merge scenario and the impact of surrounding vehicles (connected and unconnected) on the merging vehicle.

In order to perform a collision free merge, a certain safety distance between the merging vehicle and the other vehicles should be enforced. However, a problem arises when there is not a sufficient gap between vehicles on the merging lane for the merging vehicle. In this situation there are a few possible options for the merge to occur: the merging vehicle could decelerate or stop completely to avoid collision, then wait for a an ideal gap to merge. On the other hand, vehicles on the main lane could slow down, speed up or even merge onto a third lane if possible generating ample merging space. Therefore, a lane merge coordination algorithm is needed to perform actions on merging vehicles providing successful and safe merges [8].

In [9], a simulation of a lane merge scenario was carried out, accompanied by a pattern recognition model for decision-making. The model consisted of a nine grid cell, in which each cell is marked as empty or occupied according to the information received from neighbouring vehicles. The trajectory of the merging vehicle was fitted into a 5° polynomial function. The simulation provides different trajectory models using different actions such as acceleration, deceleration and wait to command the vehicle into the optimal merging point.

Function models to evaluate decision on lane merge have also been explored and built. In [10], a low-complexity lane merge algorithm was presented, determining the viability of a lane change manoeuvre, where a gap and time slot is selected to preform the lane merge. The mathematical models that calculate longitudinal and lateral control trajectory, depended on weightings to reach its optimal behaviour. However, the authors reiterated the need of a dynamic prediction model in situations where the trajectory was aborted or the scenario did not match the optimal weighting given.

A similar approach was used in [11], where the focus was to ensure optimal road space for when the lane change occurs. Thus, a two lane road was divided into cells, that can be empty or containing a vehicle, with the applicable four different actions for vehicles: acceleration, deceleration, random action, and maintaining vehicle motion. Three types of lane changes were investigated: *tail to head*, *head to tail*, and *random*. The *tail to head* approach resulted in superior results when being compared to a random lane merge. The approach assumes that all vehicles on the road are connected, such that they can communicate among themselves. Differing between a realistic scenario where the time and gaps between the vehicles are essential for a lane merge, as well as the connectivity of all the vehicles on the road.

In [12], the authors presented a collision prevention system based on fuzzy logic control to update the acceleration or deceleration of vehicles, in order to improve the lane change safety. The system performance was tested using simulated relative distance and speed measured from a radar. The algorithm assumes that the merging vehicle can merge and only provides acceleration control to avoid collisions with the vehicle in front of it. Furthermore, [13] a simulation was performed to obtain the desired steering angle and longitudinal acceleration to maintain an automated driving vehicle. In this case a stochastic model-predictive control was used, however behaviour of surrounding vehicles was predicted by means of a probabilistic collision detection.

Another approached tackling lane merging utilised a representation of the on-road environment (Dynamic Probabilistic Drivability Map), presented in [14]. The automotive test bed included cameras, radars and lidar sensing. Delivering cost effective recommendations based on dynamic programming. The theoretical formulation of this work was tested with data from 40 real-world merges. Although the approach is considered early stage. In [15], the authors consider a transition stage in the path to fully autonomous transport, with mixed-autonomy driving. In the study, a mixed-autonomy driving is considered as a collaboration of vehicles resembling a Nashor Equilibrium state, to ensure that the collective reward for lane merging is optimal. The approach simulated the role of a driver via a keyboard.

B. Reinforcement and Deep Learning for lane merge

Using machine learning is by no means a new approach to tackle lane merge prediction. In [16] the authors suggested the use of dynamic probabilistic drivability map. The problem was framed in an optimisation scope, reducing the cost of a lane change and merge scenario. The model was testing against free-flow and dense traffic, in a real world scenario, providing an insight in the adaptability of the model in the current status of the road. Although, as expected the different scenarios brought by sparse environmental conditions for the model to deal with, which is in need of fine-tuning to the specific lane merge scope the model operates in.

There are a few studies applying RL or other learning techniques to connected vehicles. In [17], a work-in-progress for an on-ramp merge driving policy using Long Short-Term Memeory (LSTM) architecture with Deep Q-Learning was presented. The scenario considers an on-ramp merging involving three vehicles: the merging vehicle and two vehicles on the mainline. A total of 9 variables are used: for the merging vehicle, 5 variables describe its driving state (speed, position, heading angle, and distances to the right and left lane). For the other two vehicles, only speeds and positions are known. The algorithm has not been verified or validated.

A deep *RL* approach was adopted in [18], handling input values from camera and laser sensor the vehicle owns with an embedded GPU for decision making. The work also proposes a monolith architecture embedded in the vehicle, and does not consider a micro service approach in which the connected and unconnected vehicles can co-exist in the same scenario. Lastly, every vehicle that includes the equipment specified would have to generate the calculations multiple times, making it more costly albeit running in real-time.

Research has also been carried out about the functionality and challenges of incorporating Deep Learning into vehicles. [19] discusses the strict assessment that needs to be undertaken before the use of Deep Learning can be considered and commercialised in autonomous vehicles. Challenges included dataset completeness, Neural Network implementation and the transfer of learning. These challenges remain to this day and are great hurdles when designing and implementing a Deep Learning approach for vehicles. It is clear that reproducing similar results in different environments from research papers seldom work. Both intrinsic (e.g. hyper-parameters) and extrinsic factors (e.g. environment) influence the performance of the agent albeit using the same approach that papers have undertaken. Therefore, suggesting that the use of *RL* is experimental and relative to the scenario that the agent operates as also seen by [20].

Other approaches propose the use of Artificial Intelligence (AI) for Lane Changing excluding the application deep learning, As seen in [21]. Where the authors propose the use of Bayes Classifiers and Decision Trees to predict whether or not a vehicle can merge. Although the results proved accurate in some cases, the approach did not detail the lane merge trajectory that should be used to execute such a manoeuvre that would result in a safe lane merge. Hinting that the use of non deep-learning AI may not capture all the environmental states necessary to learn and produce a safe trajectory for a lane merge scenario.

There are some challenges when applying RL to deploy a Lane Merge Coordination system. RL is the study of the agent's ability to interact with an environment whilst accumulating the highest reward possible. Balancing the phases of exploration and exploitation with respect to the accumulation of rewards is difficult to determine. This problem is uniquely unidentified with *RL* and does not necessarily apply to other learning branches. The exploration phase, cannot be summarised or terminated quickly, as this would impact the safety of the road and human lives. In a critical scenario such as a lane merge, it is of vital importance that the total safety of the merge and road is the main goal, and hence the learning phase may be prolonged compared to other applications of RL (e.g., gaming or stocks), in order ensure such an outcome is obtained. The Bellman equation [22] only guarantees a convergence for an optimal value function if every state is visited infinite number of times. A large amount of data would be needed in order to simulate visiting the states an infinite number of times, obviously infeasible due to time constrains. However, with some work done in function approximation, the reward can be calculated with the state and action only, thus minimising the storage needed to hold the infinite sets of combinations, which is where the deep learning component alleviates the problem by generalising the approximation function. Reward function also provides a way for the agent to prioritise tasks codifying necessary behaviour to an agent, which is vital for an optimal function to be learned [23]. In terms of the RL architecture and implementation, this paper focuses on the design and performance of the reward function.

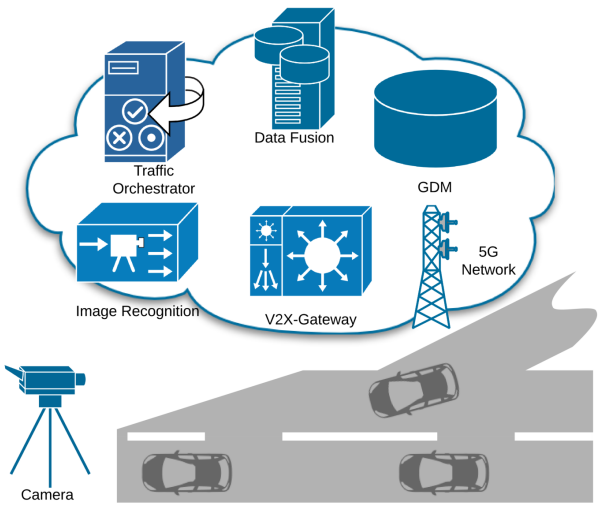


Fig. 1. In the lane merge coordination scenario, a connected vehicle attempts to merge onto the main lane while the edge cloud coordination system determines and sends trajectory recommendations to connected vehicles.

III. Architecture and System Model

This section provides the high-level architecture and the system model for a centralised coordination system. This system plans the trajectories for connected vehicles on the road to ensure sufficient space for a merging vehicle.

A. Lane merge coordination

The lane merge scenario examined in this work is depicted in Fig. 1. A connected vehicle will attempt to merge onto a main lane where connected and unconnected vehicles are present. Through an edge cloud approach, bespoke trajectory recommendations are determined and sent by central coordination mechanism to connected vehicles. Five distinct components facilitate the lane merge coordination: a *V2X Gateway*, an *Image Recognition* system, a *GDM*, a *DF* and a *TO*.

The *V2X Gateway* is responsible for forwarding messages to the various applications and interfaces in the architecture. The *V2X Gateway* acts as a communication medium that connects the interfaces and applications to connected vehicles based on a publish-subscribe message exchanging approach. This method of communication occurs across a mobile network. Applications must subscribe to the *V2X Gateway* to receive

TABLE I
MESSAGES USED IN THE LANE MERGE COORDINATION.

Message	Description
Road User Description	A Road User Description (RUD) contains the information relating to a specific vehicle. This is the information that is captured by the *Image Recognition* system or sent directly by the connected vehicle encompassing trajectory and localisation data. This is used to sync the real environment with the architecture.
Notification	A notification is used, to established a subscription and on-going messaging service with the orchestrator. Notification messages are sent from the *Global Dynamic Map (GDM)* to the *Vehicle-to-Everything (V2X Gateway)* for synchronisation and onto the *TO*. The notification messages include the crucial information that is required to provide trajectory recommendations not limited to but containing RUD of all vehicles.
Manoeuvre Recommendation	A manoeuvre recommendation is used to send trajectory recommendations. A trajectory recommendation contains the individual points constituting to a manoeuvre that a road user should execute.
Manoeuvre Feedback	A manoeuvre feedback provides the traffic orchestrator with the response in order to check if the manoeuvre recommendation is accepted, rejected or aborted.
Subscription Request	The first step of messaging process is to form a subscription request message. After which the *TO* can start receiving RUDs.
Subscription Response	Once the *V2X Gateway* receives a subscription request, a response is made by the *V2X Gateway* back to the *TO*.
Unsubscription Request	This is sent from the *TO* to the *V2X Gateway* to unsubscribe from the notification service.
Unsubscription Response	The purpose of this is to confirm a successful disconnection from the notification messages.
KPI message	This message is sent from each component every time it receives or sends a message. These messages are received by the KPI evaluation platform to calculate the efficiency of the system.

messages about vehicular features and trajectory information. The mobile network seeks to maintain a set of baseline requirements.

An *Image Recognition* system [24] collects information about all the vehicles on the road in a specified area. This information includes the localisation and trajectory-based parameters attributed to a specific road user RUD. Information about connected and unconnected vehicles are collected and processed sending all the information to the V2X Gateway, which in turn forwards the messages to the *GDM*. The *GDM* stores environmental information about connected and unconnected vehicles in a database. This information is delivered from the *V2X Gateway* system. The *GDM* ensures that stored RUDs are up to date.

The *DF* provides a synchronisation mechanism for RUDs originating from different sources (e.g., one from the *Image Recognition* system and a connected vehicle in a closely localised time frame, respectively). The *DF* updates the information in the *GDM* to be dispatched to applications that are subscribed to a specific location boundary.

The *TO* will store and process environmental factors about connected and unconnected vehicles to give rise to trajectories for connected vehicles. The *TO* needs to consider time-critical variables such as the timestamp of the vehicle location, the speed of the vehicle and the vehicle-specific dimensions. The *TO* provides a coordinated trajectory recommendation for a single or set of road users, which will then be sent to the connected vehicles through the *V2X Gateway*. The connected vehicles have the choice to either accept, reject or abort the recommendation. This feedback information is supplied by the connected vehicles to the *V2X Gateway*. The feedback can be used to recalculate trajectory recommendations.

Unconnected vehicles are not able to communicate with the *TO* and they cannot interpret or use trajectory recommendations. However the lane merge coordination is aware of unconnected vehicles by means of the *Image Recognition* system. This *Image Recognition* system provides the *GDM* with the RUDs to be stored. The road user information will be requested by the *TO* to create trajectory recommendations. To this end, a set of messages need to be defined for communicating all the components within the lane merge coordination. Messages used in the communication will employ a common message formatting based on JavaScript Object Notation (JSON). This allows to communicate human-readable text, that can be received and processed in any software component. The message implementation for all messages uses JSON as specified in [25] and a description of the messages is presented in Table I.

In this paper, we will focus on the design, implementation and evaluation of the *TO* and the *DF*. As the other components: *Image Recognition* system [24], *V2X Gateway* [26] and *GDM* have been created and explored by external work [6]. The proposed architecture model follows a microservices approach, consisting of a suit of independent components that can communicate using Transmission Control Protocol (TCP) or User Datagram Protocol (UDP). So long as the connections and the message format are guaranteed, the TO is not affected by the interaction of other components. However, due to its nature, the DF is dependent on the quality of the data it receives from other components.

B. Traffic Orchestrator

The *TO* must demonstrate a level of safety concern and overall reliability. To do so, it must implement certain functionalities to appropriately integrate with the environment and perform a successful lane merge. These functionalities must provide an implementation such that the *TO* can:

- Provide safe manoeuvre recommendations that seek to reduce collision.
- Provide insightful explanations during the execution of the *TO*.
- Successfully establish a subscription service with the *V2X Gateway*.
- Receive and send data to the *V2X Gateway* over a TCP connection.

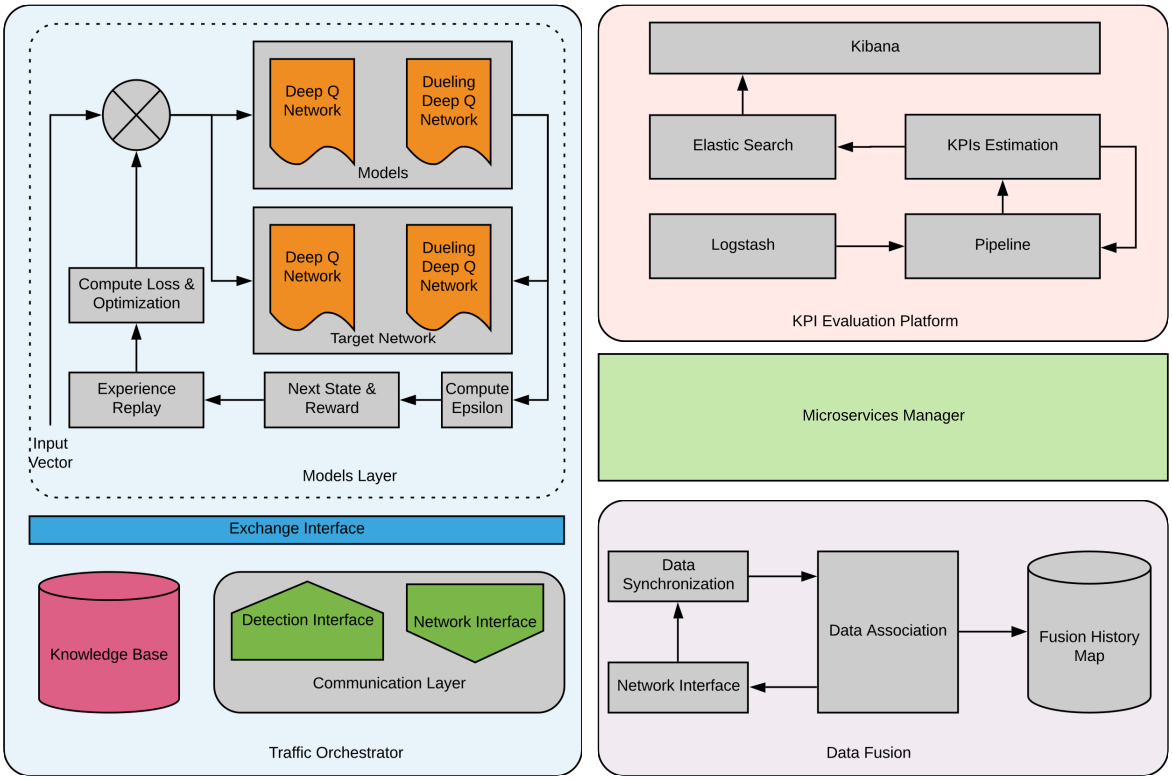


Fig. 2. Proposed *TO* architecture.

- Handle JSON messages to process subscription responses, manoeuvre feedback and RUDs.
- Calculate manoeuvre recommendations based on the information provided by the *GDM*.
- Process the feedback of a specific manoeuvre recommendation.
- Successfully merge a connected vehicle onto a single or multi-lane carriageway.

The proposed architecture for the *TO* system is presented in Fig. 2. The main purpose of the *Detection Interface* is to, receive any data being sent, over a TCP connection, from the *V2X Gateway*. The *Detection Interface* also acts as an intermediate filter that will read JSON strings and process the JSON messages into more compact and efficient structure to be used by the TO. Similarly, the *Network Interface* will act as a filter that will convert and translate information within the *TO*, to JSON messages to be fed into the *V2X Gateway*.

A *Knowledge Base* has been designed to store the information sent to the *TO*, so that it can independently work and will not be influenced by any unusual behaviour of other components. It is maintained to guarantee that a manoeuvre recommendation is calculated based on all current road-environment knowledge, containing only the RUDs that the *GDM* has most recently transmitted. The *Knowledge Base*, mimics the access and modification functions of a typical database. As such, the knowledge base is able to:

- Insert a RUD to represent the most recent environmental snapshot.
- Provide access to RUDs being stored in order to query certain conditions and provide manoeuvre recommendations.
- Delete RUDs if they are out-of-date or no longer relevant.
- Provide a search function allowing the retrieval of a RUD by their Universally Unique Identifier (UUID).

The *Exchange Interface* has been designed to have two responsibilities: execute the *TO* application and mediate the flow of information across all interfaces in the *TO*. The *Exchange Interface* takes structured data from the *Detection Interface* and appropriately forms the data into entities. These entities can then be reused throughout the rest of the system in a consistent manner. This component directly interfaces with the *Knowledge Base* and will collect structured RUDs. Another functionality of the *Exchange Interface* is to provide access for consistent communication methods to a set of *TO* functionalities, allowing different algorithms to run on top of it.

There are three major design factors that need to be considered with great detail, when proposing a lane change.

- Safety distance from all cars on the highway; this is to ensure that the cars keep the safety breaking distance at all times.
- Positioning and acceleration of the connected vehicle in comparison with the values of other connected vehicles on the road. Therefore, the motivation of the TO, is to provide positional coordinates as well as acceleration and

speed values to the connected vehicle to give a path to follow for a merge.
- Acceleration and positional values of the neighbouring vehicles, need to be stored. TO passes instructions to other connected vehicles creating a multi-agent solution that benefits the interest of every vehicle on the road.

In order for the *TO* to communicate with the other components in the stack, a microservice approach is adopted ensuring, cross platform compatibility and hence the ability to communicate with ease. The *TO* adopted *jit* trace files, to provide a way for tensor manipulation allowing the python created model to run on c++, maintaining the efficiency and performance known to c++ on the *TO*. Functionality of the *TO* is broken down into 5 main essential components.

- Sending or receiving a subscription to and from the *V2X Gateway*.
- *TO* is required to receive the notification messages from *V2X Gateway*. This is a key step, as the messages contain the vehicle descriptors that act as input feature vectors for the *RL* model.
- With the information taken from the notification messages, the *TO* needs to send the manoeuvre recommendation of the connected vehicle back to the *V2X Gateway*, forwarding it to the vehicle in need.
- Manoeuvre feedback is vital to the *RL* learning process, as it facilitates its reward, whereby a successful merge correctly reinforces the algorithm and increases the accuracy and performance.
- To terminate the connection of the *TO* and the *V2X Gateway*, an unsubscribing mechanism needs to take place.

C. Data Fusion

RUDs from both the connected vehicles and the *Image Recognition* system are received in the *DF*. This component is responsible for updating the *GDM* data with the latest RUD and avoids having a duplicated RUDs coming from a connected vehicle and from the *Image Recognition* system. Also, it is responsible for enhancing the RUDs accuracy by combining most accurate values from each sources, i.e the acceleration from the *Image Recognition* system is less precise than the one provided by the connected vehicle. Due to its nature, the DF is highly dependent on 2 factors: the quality of the data sent by the *Image Recognition* system and the rate of the messages coming from the *V2X Gateway*. The *data fusion* consists of four different components:

1) *Network Interface* is responsible for communicating with other components. It receives RUDs from the *V2X Gateway*, deserialises the and forwards messages to the *Data Synchronisation*. Also, it receives the fused and corrected RUDs from the *Data Association*, serialises and sends them to the *GDM*.
2) The purpose of *Data Synchronisation* is to synchronise received RUD in time. It collects all received RUDs during a certain period. The period was set to $100ms$ corresponding to half the frequency of incoming messages. It then extrapolates each one to the same temporal reference. Given that the period is small considering the speed of the objects a uniformly accelerated rectilinear motion was found sufficient. Moreover, it updates the positioning information of each object and its timestamp.
3) *Data Association* matches objects detected by the camera with connected vehicles. For each detected object by the camera, a check is performed to verify if the object is already matched with another in the *Fusion History Map*. Then, it can raise or lower the confidence according to the Euclidean distance and the angle between them.
4) *Fusion History Map* stores the history of matched objects. For each matched object, the following information is stored: last seen timestamp, connected vehicles UUID, camera detected UUID and confidence level. The history map is cleaned every few seconds based on the last seen timestamp.

D. KPI Evaluation Platform and Micro-Services Manager

The main goal of the KPI Evaluation Platform is monitoring the overall system in real time by aggregating every component logs in a single and easy to search platform. This platform allows the evaluation of software and network KPIs. In terms of performance, the main KPI of this component is the *trajectory delivery time* which provides the time it takes to deliver a manoeuvre recommendation, since the moment a RUD is first sent. Additionally, reliability is the second KPI measured by the messages loss of each component. The KPI Evaluation Platform provides three main functionalities:

1) Collect received logs from different components by exposing a network communication interface. The received messages are parsed, formatted and enriched, then forwarded to the database.
2) A database is used to store the messages and offers a query language to explore the data and compute KPIs.
3) A Graphical User Interface (GUI) for data visualisation that allows users to monitor data in real time. It offers the ability to explore raw data and to create charts and dashboards.

The *Elastic stack* is a set of well integrated open source software components designed for this purpose: *Logstash* for data collection, *Elasticsearch* to store and to query data and *Kibana* for data visualisation. The Micro-Services Manager enables connectivity across independent components in a scalable solution, providing a central logging system where all the components can be monitored for further manipulation and analysis.

IV. Reinforcement Learning Model

This section describes the decisions taken regarding the design of the *RL* models. Furthermore, it briefly describes the dataset utilized for training the algorithms.

A. Training Dataset Exploration and Manipulation

Two different datasets collected by Federal Highway Administration Research and Technology - Coordinating, Devel-

oping, and Delivering Highway Transportation Innovations[1] were used in this work. The datasets represent the data collected on two American motorways: Interstate 80 Freeway ($I-80$) and $US-101$.

From this dataset, we extracted every lane merge scenario with 3 vehicles, where the coordinates, speed, acceleration, size and heading of the vehicles are considered. The lane merge scenario itself can be described with 3 vehicles, whereby, the merging vehicle has the goal of joining a new lane in between a preceding and following vehicle. Thus the adoption of 3 vehicles in this work, reflects a lane merge instance, by considering a following, preceding and merging vehicle. Limiting the scenario in this way, allows for a more compact implementation that is conservative on both computing resources and model convergence time. Moreover, this processes can be repeated through the entire number of lanes allowing for a dynamic lane merge scenario regardless of the amount of lanes on the road. Further reducing the need for lane and road information, as the merging vehicle needs only to concern itself by aligning its position corresponding to the line formed by the preceding and following vehicle in the target lane. In the likely event where there is no preceding or following vehicle present, a mock vehicle is procured to simulate the merging scenario.

The safety conditions of a lane merge are based on the intersection of two pairs of circles. Firstly, one circle has the central x and y coordinates of the merging vehicle and the radius of 10% of its speed in $\frac{km}{h}$. This is compared with a second circle which has the central x and y coordinates of the preceding vehicle and the radius of its length. If the circles do not intersect then the data is viable for the RL algorithm to train on.

B. Recommended actions

Design premises for detecting lane changes and data labelling are based on [27]. However, several changes had to be made due to the use of merging actions for the deep reinforcement learning. The *TO* utilises the recommended actions obtained by the deep reinforcement models to apply Newtonian equations in order to calculate the respective speed, acceleration, distance and position for each vehicle on the road. The actions are thus configurable and can be optimised to remove positional and speed bias in different merging scenarios. Defining the actions is vital to ensure the neural network layers are configured correctly and as such obtain an accurate model. The actions that can be given to a connected vehicle is broken down to:

- Acceleration: Increases the acceleration
- Deceleration: Decreases the acceleration
- Left: Change the heading westwards
- Right: Change the heading eastwards
- auto-pilot: maintain current values.

C. RL Design

Unlike most papers that require additional layers to obtain data from the environment, this work has omitted such layers.

[1]https://www.fhwa.dot.gov

TABLE II
LAYERS COMPOSING THE VARIOUS RL MODELS

	Architecture	Type	Input (nxd)	Output (nxd)
DQN	Vanilla	Linear	$1x19$	$19x200$
		RelU	$19x200$	$19x200$
		Linear	$19x200$	$1x5$
Dueling DQN	Features	Linear	$1x19$	$19x200$
		RelU	-	-
	Advantage	Linear	$1x19$	$19x200$
		RelU	-	-
		Linear	$19x200$	$1x5$
	Value	Linear	$1x19$	$19x200$
		RelU	-	-
		Linear	$19x200$	$1x5$

This is due to the fact that the image processing is handled by lower layers of the stack. As a result data received in transformed from JSON format into a feature array including positions, speed and acceleration, which is fed directly into the model saving computational resources. The layers of the reinforcement learning model are vital to the performance of the connected vehicle. The larger the number of layers, the more the number of connections are being made from the input to output data. Three risks were identified and properly addressed: over fitting leading to trivial generalisation to new scenarios, the training time due to a large number of layers and the computational resources required and the representation of roads in the algorithm.

This work omits the portrayal of road information such as the length and width of the road to the longitude and latitude of the lane positions. This design choice, of excluding the road information allows a better generalisation of the model to other scenarios where the road information may vary significantly. Moreover, this reduces the complexity of the model by minimising the dimensionality of the input to the algorithm. Therefore, the layers of the *RL* models were designed as it is shown in the Table II which outlines the neural network structure for the Dueling DQN and DQN. The input given by an (nxd) array, represents the features obtained by the three aforementioned vehicles. The input features are mapped to 200 links through a linear function. The Rectified Linear Unit (RelU) is used as an activation function to aid the decision process of the *RL*, it is widely used in deep learning. Finally, the network outputs a (nxd) array containing the chosen action out of the 5 actions mentioned in IV-B.

On the other hand, the lower section of Table II exposes the inner workings of the Dueling DQN model. An immediate observation is the larger number of layers that the model possesses compared to the DQN model. Another observation is clear where each of the intermediate layers Features, Advantages and Value mimic the neural structure of the DQN in the sense that they are made up on 2 Linear layers and a RelU layer. The addition of these layers allows for feature-value mapping that enables each action to correspond with a cost in the environment, and the advantage obtained to the vehicle from taking said action. However, due to the increase in complexity and size of the neural structure, the training time

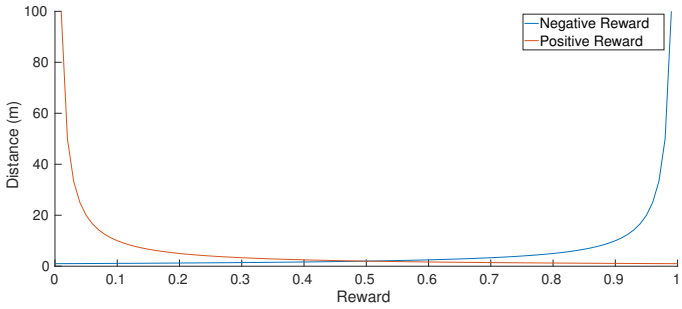


Fig. 3. Graph showing the different approaches on the assignment of rewards with respect to distance of vehicle from merging point.

increases proportionally to the number of neurons presented in each layer. As such, to alleviate this caveat, the hidden layers size of the neural layers have been reduced to negate larger training times.

D. Rewards, Loss Function and Exploration of state space

1) Reward Function: The rewards that reinforce the merging connected vehicle is calculated based on the vehicle position relative to the optimal merging point between the preceding and following vehicle. A desired terminating state, is when the merging vehicle is successfully placed in between the following and preceding vehicle, whilst maintaining an appropriate safety distance. The use of a parabolic function as a reference to the rewards assignment allows the gentle guiding of the vehicle to the optimum state through constant incremental rewards. Using a other functions may lead to unexpected behaviour of the vehicle, this is highlighted with the use of a step function, where the vehicle only receives a distinctive reward at the end of the trajectory leaving all the intermediate way points unassigned.

The aim of the algorithms is to codify the merging vehicle to mimic human-like merging scenarios, by applying minimal acceleration and changes to the overall merging scenario. At the same time, the merging vehicle should also maintain a safety distance as large as possible from the following and preceding vehicle to ensure the safety of the merging scenario. The assignment of rewards is clearly seen in Fig. 3, where the negative and positive rewards an agent can obtain follow a parabolic curve. This guides the *RL* agent to reduce the distance between the target merging point and vehicle location to obtain the greatest possible reward.

The positive (equation 1) and negative (equation 2) reward allocation denoted as r_p and r_n respectively, is calculated as follows, where d_{mp} corresponds to the distance to merging point, s is the speed and acc the acceleration:

$$r_p = \left| \left(1 \times 10^{-3} \times \frac{1}{d_{mp}} \times \frac{1}{s} \times \frac{1}{|acc|} \right) \right| \tag{1}$$

$$r_n = -\left(1 + r_p\right) \tag{2}$$

Where the reward is bounded between [−1,0] and [0,1] for the negative and positive reward function respectively.

The positive reward function was selected based on the data extracted from datasets mentioned in IV-A, as such a standardisation factor of 1×10^{-3} is employed. The reward function, focuses on assigning a reward relative to the agents distance to the merging point placed between the following and preceding vehicle. Whilst, enforcing the use of slow speeds and steady acceleration adhering to road safety precautions, to obtain a maximum reward. The negative reward function, was developed to utilise the same thought process as the positive reward function, albeit adopt a negative reward allocation scheme. More importance could be placed on the different aspects of the reward function, by introducing a weighting factor. This comes at the cost of neglecting the other merging factors of a vehicle in lane merge.

The positive reward reinforces the agent to reduce the distance to the target merging point to obtain the largest reward, whilst the negative reward punishes the agent less as the distance to the merging point decreases. The difference of two reward assignment functions greatly effects the behaviour of the vehicle during a merge. The negative reward assignment will calculate the distance to the merging point and accordingly reinforce the agent, this emphasises the agents urgency to obtain the least most negative reward as quickly as possible in order to improve the overall utility of the merging instance, hence importance is placed on the time taken to complete the merge as well as the distance the agent maintained to the merging point. Whereas, in the positive reward assignment instance, the agent receives a positive reward increasing as the distance to the merging point decreases, thus the vehicle can instead focus on accumulating the largest possible reward by mimicking a human merge with respect to safety and adaptability.

2) Loss Function: There are numerous viable loss functions that can be adopted to a *RL* approach [28]–[32] showcase ranging loss functions such as various actor-critic loss functions to fuzzy network loss functions and even bespoke loss functions utilised in [33] which incorporated probabilistic functionality. The loss function enforced greatly impacts the performance of the agent. However, Mean Squared Error was adopted in this work. As seen by the following formula:

$$loss(x, y) = \begin{cases} 0.5 \times (x - y)^2, & \text{if} |x - y| < 1 \\ |x - y| - 0.5, & \text{otherwise} \end{cases} \tag{3}$$

The loss function was specifically chosen due to its less sensitivity to outliers and exploding gradients [34]. Which would present an issue in this work since, according to section II, the measurements captured in the data set, contained many outliers, and as such the loss function provides a way to combat such faults with the data.

3) Exploration vs Exploitation: *RL* generally suffers from balancing exploitation vs exploration (see section II). The use of epsilon greedy method alleviates this problem, setting a high epsilon value at the start of training is beneficial, as a high epsilon corresponds to a higher probability of a random action, such that at the start of training the *RL* agent gets the maximum number of random actions made to explore the environment state. As the model learns the probability of a

random action is reduced as the model computes the optimum action to take. This ensures, that the model has a gradual approach to exploitation whilst still employing random actions to uncover hidden rewards that have not been seen by the model.

V. Implementation

A microservice approach was selected for implementing the architecture. It supports real world production architecture where each use case can be handled by a different application with a standardised way to access data. In this sense, the *V2X Gateway* manages the data access of each microservice. Microservices can be deployed, tested and deployed independently from one another with minimal to zero effects on other aspects of the stack. Microservice architecture also allow horizontal scaling of single components on the fly for example during rush hours. This work focuses on four components: TO, *Data Fusion*, *Key Performance Index evaluation platform* and *Microservices Manager*. The use of microservices approach allowed for the isolation of components giving the architecture more resilience in a real world scenario where one point of failure does not mean the failure of the entire architecture. Furthermore, this approach encourages scalability and flexibility of the components, thus new advancements can be applied on the architecture without minimal to zero effects on other aspects of the stack.

A. Traffic Orchestrator

The implementation of the TO allows us to evaluate different RL models and select the most optimal one. A summary of the decision making process can be seen from the pseudo-code shown in Algorithm 1 and 2. Algorithm 1 describes the process of predicting a recommendation for merging using the notification messages received from the *V2X Gateway*, either adding or removing vehicles from the knowledge base. The vehicles from the knowledge base are filtered to obtain the merging, the following and preceding vehicle, in order to pass these values as an nxd feature vector as input for the RL, and then to compute the most optimal action to undertake. This process is repeated every $100ms$ to ensure maximum road relevance as well as to keep environmental information for the road updated. Once a notification message is received containing a vehicle in need of a merge, the TO sends a trajectory recommendation consisting of one way point for the connected vehicle to undertake, the process is repeated until the notification messages no longer indicates a vehicle in need of merge. This aids the real-time reasoning of merge creation to make use of the staggering speeds of $5G$ and ensuring a safe and revised manoeuvre recommendation.

As shown in Algorithm 2, the way points are calculated from Newtonian equations, as the output of the RL merely dictates which of the actions to undertake, but not the coordinates, speed or accuracy the vehicle should achieve. The way-points are then parsed into a Manoeuvre Recommendation message format to be sent to the *V2X Gateway* and onto the vehicle for implementation. The vehicle responds with a Manoeuvre Feedback depending on the current vehicle speed, acceleration

```
Data: Vehicle Descriptors
for vehicle in VehicleDescriptors do
    if vehicle in merging lane then
        preceding_vehicle ←
          preceding_vehicle(merging_vehicle)
        following_vehicle ←
          following_vehicle(merging_vehicle)
        action_A ←
          (DuelingDQN, DQN)(preceding_vehicle,
          merging_vehicle,following_vehicle)
        way-point_WP ← Newtonian_features(action_A)
        Manoeuvre_Recommendation ← way-point_WP
        return Manoeuvre_Recommendation
    end
end
```

Algorithm 1: $Compute\_Recommendation()$

```
Data: Continuous environmental information received from
      Architecture. 100ms approx.
initialise database()
while input stream do
    message ← input stream
    if message.type notification then
        update_database()
        Compute_Recommendation()
    end
    if message.type Trajectory then
        if Trajectory rejected then
            Check_Road_Availability(database.vehicles)
            Compute_Recommendation()
        end
        update_database()
    end
end
```

Algorithm 2: $TO\_Create\_Trajectory()$

and time taken to implement the sent way-points. If the way-point is rejected, the TO recalculates another way-points based on the updated environmental information to provide another recommendation for the vehicle to complete a safe manoeuvre.

B. Data Fusion

The *data synchronisation* aligns input received from the connected vehicles and from the camera system during a period of time to a fixed timestamp by extrapolating the position of each road user. The *data association* take the extrapolated positions of the road users and tries to match each connected vehicle with a camera detected object. It uses the euclidean distance between two objects to determine if they are the same road user. In some cases, due to measurement inaccuracies the distance can be higher than the width of a lane, this is corrected by calculating the angle between the heading of the following object and the preceding one. Distance and heading thresholds were adjusted to match the precision of the incoming data. Each time two objects are fused the confidence of the match is increased in the *fusion history map*. On the contrary if two object are not fused the confidence is lowered. If an object was not fused during the

```
Data: camera system detected objects
Data: connected vehicle objects
for camera_object of camera system detected objects do
    for connected_car of connected vehicles objects do
        distance ← euclidean distance(camera_object,
          connected_car)
        if distance ≤ LOWER_DISTANCE_THRESHOLD
        then
            fuse(camera_object, connected_car)
            raise confidence(camera_object, connected_car)
        end
        else if distance ≤ HIGHER_DISTANCE_THRESHOLD
        then
            angle ← angle between two
              objects(camera_object, connected_car)
              if |angle − connected_car| ≤
              HEADING_THRESHOLD then
                fuse(camera_object, connected_car)
            end
            else
                lower confidence(camera_object,
                  connected_car)
            end
        end
        else
            lower confidence(camera_object, connected_car)
        end
    end
    if not fused then
        matched ←
          fusion_history_map.find(camera_object.uuid)
          if matched AND matched.confidence ≥
          CONFIDENCE_THRESHOLD then
            fuse(camera_object, connected_car)
        end
    end
end
```

Algorithm 3: data association algorithm

association phase the algorithm checks in the *fusion history map* if it has already been matched with a high enough confidence and fuses the objects accordingly. A description of this algorithm in pseudo code is available in Algorithm 3.

C. KPI evaluation platform

The implementation of the KPI evaluation platform is based on two main elements: KPI Message format and *Logstash* pipeline. Fig. 4 shows an illustration of the web-based GUI where a dashboard was created for real-time monitoring.

1) KPI Message format: The message payload was designed to reduce the network overload and *Logstash* processing time. It contains the following information:

- Component: a unique identifier for each component
- Type: the message type as listed in Table I.
- Direction: identify if the message was received or sent
- Partner: the unique identifier of the partner sending or receiving the message
- Message_uuid: the unique identifier of the message
- Initial_timestamp: the timestamp of the first emission of the message
- Timestamp: the timestamp of the moment the message was sent or received by this component

```
Data: KPI message received from a component
while input stream do
    message ← input stream
      parse message as kpi log()
      if !message kpi log then
        save(message)
    end
    if message kpi log then
        message.total_duration ← message.timestamp -
          message.initial_timestamp
          if message.direction received then
            related ← find related sent log (message_uuid,
              partner)
              message.network_time ← message.timestamp -
              related.timestamp
        end
        else
            related ← find related received log (message_uuid,
              partner)
              message.compute_time ← message.timestamp -
              related.timestamp
        end
        save(message)
    end
end
```

Algorithm 4: *Logstash* pipeline

2) Logstash pipeline: With the information described in the previous section it is possible for *logstash* to calculate network and software processing times by aggregating sent and received messages. Each time a KPI message is received by the collector it is parsed. The pipeline then calculates the total duration by subtracting the initial timestamp from the timestamp field. Then the script searches for the previous event of the same message and calculates the elapsed time as described in Algorithm 4.

The *trajectory delivery time* is one of the KPI that is important to the evaluation of the communication within the trajectory creation scope. We define the *trajectory delivery time* as the time it takes to deliver a manoeuvre recommendation, since the moment a RUD is first sent by the *Image Recognition* system or a vehicle. The KPI Evaluation Platform computes the time of every RUD sent from the *Image Recognition* system (or the vehicle itself) to the *DF* and passed onto the *TO* to calculate and forward a manoeuvre recommendation to the *V2X Gateway* which broadcasts it to the vehicle.

D. Microservice Manager

A containerisation approach was used to incorporate the variety of microservices used for the lane merge coordination. This enabled minimum dependencies for implementing the services in the test track as well as scaling the solution into a production ready environment. Software components were containerized using Docker as a hypervisor, this allows to manage images, drivers and set to communicate on a single network to ensure maximum connectivity. Components deployed in the edge cloud use the Graylog Extended Log Format (GELF) logging driver. This encapsulates the standard output of a container into a GELF message, compressing

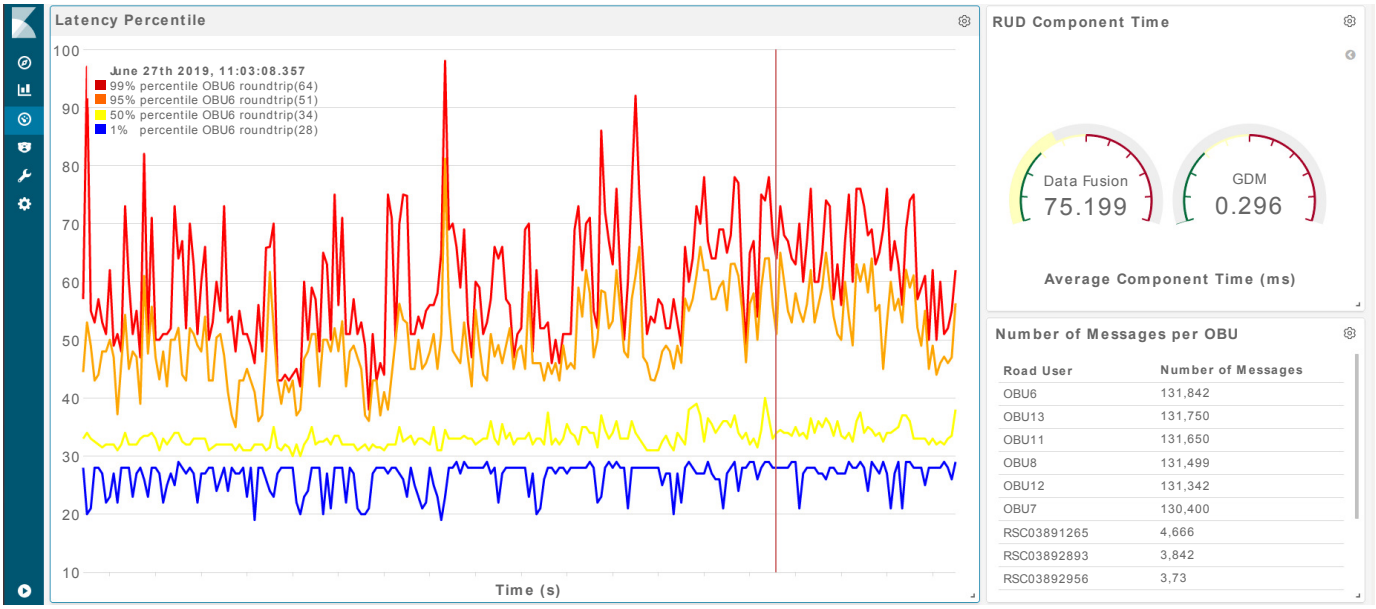


Fig. 4. Illustration of the *KPI* real-time monitoring graphical interface.

through the use of gzip and sending UDP messages to the *Logstash* server. while on board units in connected vehicles sent the messages in plain text through UDP to the *Logstash* server through the $5G$ network. Enabling a central logging system for all the components that can be monitored for further manipulation and analysis.

VI. Results

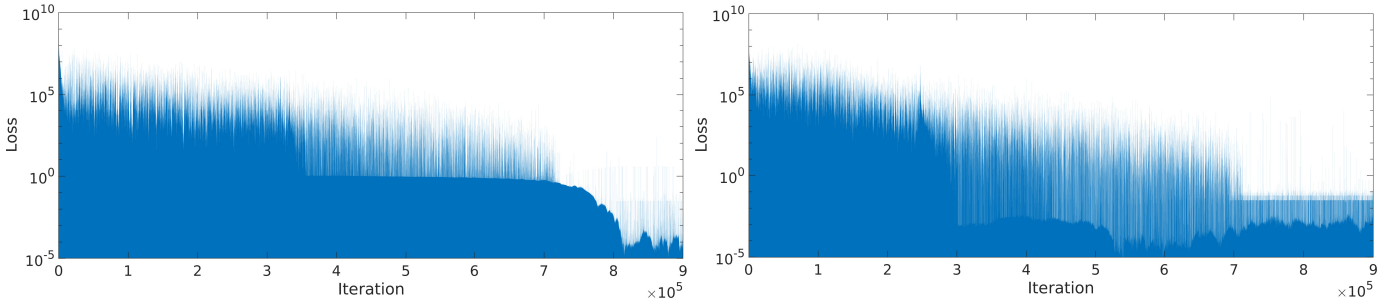
To analyse the effectiveness of the lane merge coordination solution, this paper focuses on two integral testing phases: a performance evaluation of the RL models and a set of real world tests [6] using connected vehicles. The purpose of the performance evaluation is to determine the optimal RL model to be used for the TO on real tests. For this, we have selected 4 main KPIs: loss, assignment of positive vs negative rewards, accuracy and predicted vs human trajectories. Furthermore, we include a comparison between the proposed RL models and other state-of-the-art ML models (*Gradient Boosting*, *Random Forest* and *Linear Regression*) [27]. On the other hand, the real world tests defines an actual merging scenario on a test track, in which the TO is predicting live trajectory recommendations to connected vehicles. A comparison of these predicted trajectories with recorded data from the logging microservice in the architecture is also provided. In this case, we selected 5 KPIs to analyse the lane merge coordination: predicted vs human positioning, inter-vehicular distance, merging acceleration, trajectory length and trajectory delivery time.

A. Performance Evaluation Tests

Two different RL models (i.e, DQN and Dueling DQN) were trained using the dataset described in section IV. This dataset was split into 3 subsets: training, testing and validation where each of them with 70%, 20% and 10% of the size of the original dataset respectively. The training subset contains 10^5 merging instances where each merging instance is approximately 70 data points that represent a merging scenario. The merging scenarios are randomly selected from the dataset, but the data points are iterated over chronologically to provide a logical merging instance. The model predicts and allocates a trajectory recommendation to connected vehicles for a successful merge.

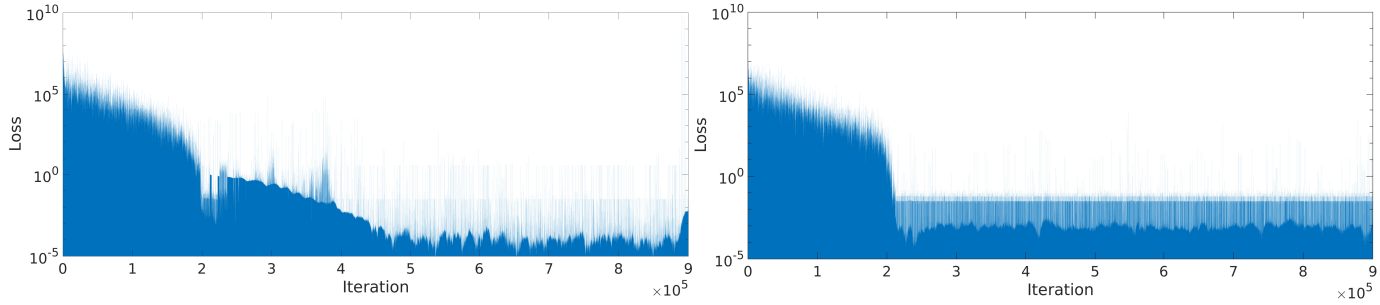
1) Loss Comparison of RL models: Fig. 5 describes the loss obtained per iteration for the DQN model. As expected, Fig. $5a$ and $5b$ show that the loss is gradually decreasing as the number of iterations grows until the threshold. The main difference when using negative or positive rewards is loss stability, since Fig. $5a$ shows a more stable behaviour compared to Fig. 5b, there is a sharp drop close to the iteration 3.5×10^5 in Fig. 5b that triggers lower loss values. Similarly, Fig. 6 present the loss for the Dueling DQN model which performs significantly better in terms of loss compared to DQN. Fig. 6a and 6b follow the same trend than in Fig. 5 correspondingly, however Dueling DQN converge sooner. This can be seen from the lowest loss obtained by the positive assignment of rewards of 10^{-5} at approximately iteration number 2.3×10^5, which is significantly lower than its counterpart in the DQN model. Furthermore, the negative reward assignment also portrayed signs of a shorter convergence time for Dueling DQN, obtaining its lowest loss at iteration 5.2×10^5. This means that Dueling DQN allows a reduction in terms of training time and computational resource consumption.

Moreover, the rewards that reinforce the agent heavily impacts loss, convergence and stability of the models. Dueling DQN produced the most stable learning phase when using positive reward, resulting in a lower loss and earlier convergence. The architecture developed for Dueling DQN (see section IV) provides better values in terms of stability and training time for the agent, which is noticeable by a smaller loss amplitude and an earlier convergence of the model. On the other hand,



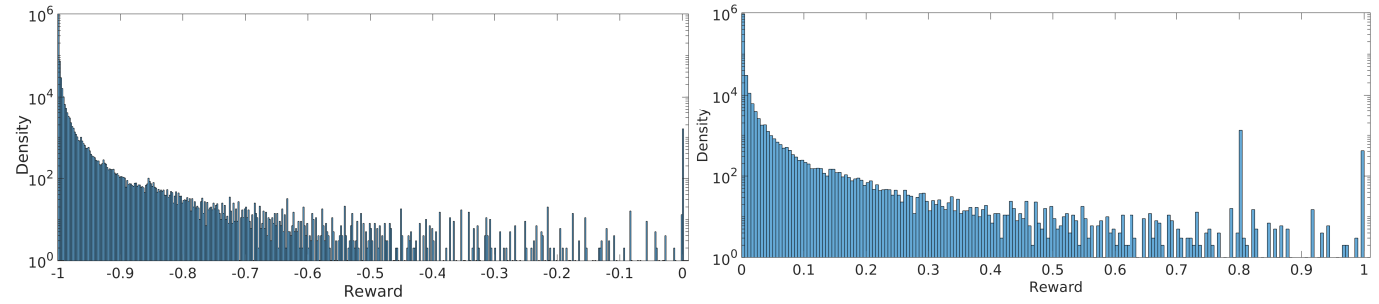
(a) Negative Rewards (b) Positive Rewards

Fig. 5. DQN Network Models Loss Comparison through negative and positive reward assignment.



(a) Negative Rewards (b) Positive Rewards

Fig. 6. Dueling DQN Network Models Loss Comparison through negative and positive reward assignment.



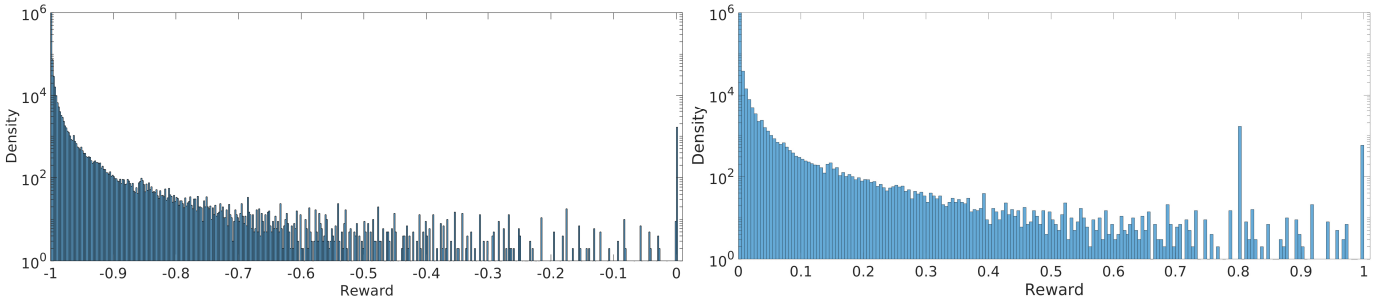
(a) Negative Rewards (b) Positive Rewards

Fig. 7. Histogram for comparing assigned rewards for trajectory recommendation by DQN agent.

the DQN model could not compete with the loss presented by the Dueling DQN even with a longer training time. This can be closely correlated with the prediction of trajectory recommendations that the model forecasts. Whereby, the lower the loss the closer the predicted manoeuvre is to the recorded manoeuvre in the dataset.

Neglecting the hardware used for training the models, the convergence of the Dueling DQN occurred at approximately 2.3×10^5 and 4.8×10^5 iteration for positive and negative reward allocation respectively, whilst the DQN occurred at 8.2×10^5 and 5.3×10^5.

2) Positive vs negative rewards assignment: To be able to fully compare the effect of the positive vs negative rewards have on the DQN model, Fig. 7 and 8, highlight the count of positive and negative rewards obtained during training time of the model. The assignment of the rewards impact the success of the trajectory recommendation. Both the models assigned the negative reward schema follow the same general shape seen in Fig. 7a and Fig. 8a, where there is an inverse proportional relation between the magnitude of the rewards and the reward obtained, until the model reaches a successful merge obtaining a reward of 0 where the density increases greatly, indicating a converged model. The density of rewards obtained at 0 is 1.7×10^4 by the Dueling DQN compared to 1.5×10^4 for DQN. On the other hand, the positive assignment of rewards provided a surprising shape for the graph, where still following the general indirect proportionality presented by the reward, the sudden increase in magnitude takes place at a reward of 0.8. Furthermore, the density of that reward surpasses the density obtained by a reward of 1 by a minuscule factor. Notwithstanding, the Dueling DQN still receives a higher density of greater rewards than the DQN varying by a density of 3×10^3. Although, the agent obtained a large



(a) Negative Rewards (b) Positive Rewards

Fig. 8. Histogram for comparing assigned rewards for trajectory recommendation by Dueling DQN agent.

density of rewards at 0.8 proves the existence of a global minima in the positive assignment of rewards that the model needed to surpass in order to obtain a successful merge that the use negative rewards did not face. Therefore, the loss and the reward assignment are used hand in hand to obtain a clearer insight of the model performance for optimal model selection.

3) Model accuracy on Training and Validation subsets: To test the generalisation of the model, with respect to the recorded dataset that the model was trained on, the test subset is used to gain insight on performance of the agent. If the generalisation for the Dueling DQN with a positive reward function is successful in the testing subset, then it can be safely assumed that it could be used to generalise to different merging scenarios in real world instances. The accuracy is based on the number of successful merges that the model predicts from the total number of merging scenario presented in the subset. The accuracy is measured by recording the predicted trajectory along with the human recorded trajectory, if the predicted trajectory reaches the merging point with the same route as the human trajectory it is considered a successful merge with very high accuracy, on the contrary when the predicted trajectory does not follow the same route as the human trajectory, it is considered as a low accuracy prediction. However, the agent can also recommend trajectories that reach the merging point successfully, but by taking another route. The goal is achieved whereby the merging vehicle enters the target lane, albeit the fact that a different route was taken. Finally, the unsuccessful merge instance is where the merging vehicle does not reach the merging point at all.

In Fig. 9, successful merges mimicking human-like trajectories during the training phase are presented. The individual points at the start of each iteration number, correspond to the accuracy obtained in both models for the training subset. A squared function is used to fit the accuracy shown, to give an approximation to the accuracy that the models had in between epochs. The results are to be expected when considering the unseen scenarios of the dataset and the uncomplex architecture of the DQN compared to that of the Dueling DQN. The testing accuracy of the models was carried out with the same criteria as the training accuracy where the most optimal model from the training run iterations was obtained for both DQN and Dueling DQN. The testing accuracy is seen in Fig. 10, the Dueling DQN model scored an accuracy of 64% for successful

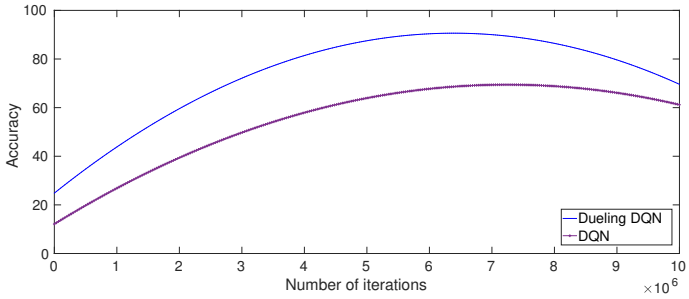


Fig. 9. Training accuracy for DQN and Dueling DQN models incorporating positive rewards. Only successful merges mimicking human-like trajectories are considered in this case.

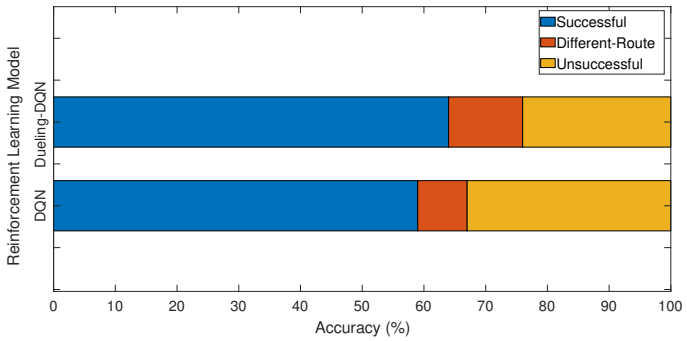


Fig. 10. Testing set accuracy of models for successful human-like trajectories, successful merges with different route and unsuccessful merges.

human-like merges and 12% for successful merges with a different route. Whereas, the DQN model scored 59% and 8% respectively. Dueling DQN has a better performance compared to DQN for human-like merges and merges with different route. However, the accuracy fitting of the model can prove to be inaccurate, but is used to represent the data to provide a general consensus on the theoretical accuracy that would be obtained mid epochs.

4) Model accuracy compared with others ML models: In order to provide some insights of the expected accuracy, the approaches taken in this paper regarding the implementation of the RL models is compared with state-of-the-art ML prediction algorithms, i.e., *Gradient Boosting*, *Random forest* and *Linear Regression*. The same Next Generation Simulation Model (NGSIM) dataset was used in all the tests, so that

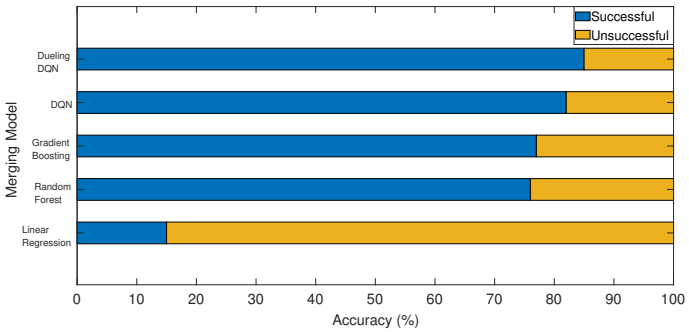


Fig. 11. Acceleration accuracy comparison between state-of-the-art ML models and the RL proposed models.

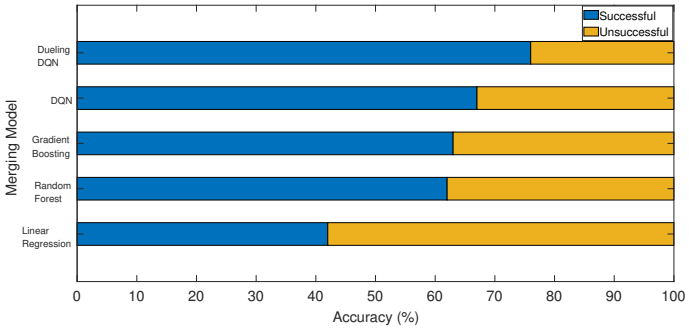
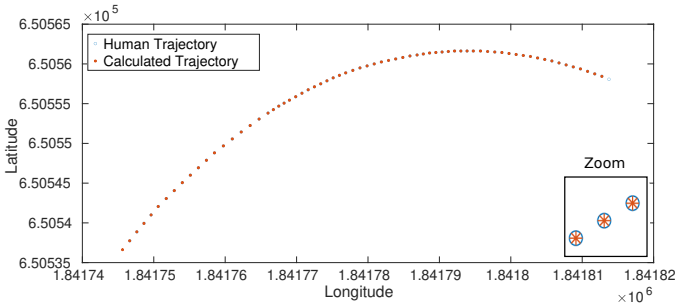
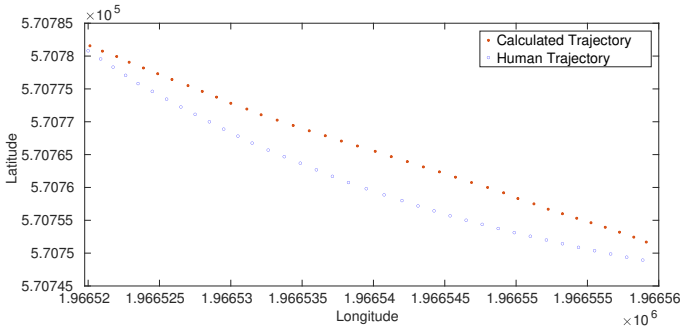


Fig. 12. Heading accuracy comparison between state-of-the-art ML models and the RL proposed models.



(a) Successful human-like merge



(b) Successful merge using a different route

Fig. 13. Difference of human vs predicted trajectories for successful merges.

the feasibility of the approach can be highlighted when set side by side. Fig. 11 and Fig. 12 depicts the merging model against the accuracy and heading obtained in the testing subset respectively. The adoption of an RL component be it DQN or Dueling DQN significantly improved the quality of the lane merge in terms of accuracy for acceleration and heading suggestions, contrasting a ML approach. It is important to note that the acceleration accuracy, for the RL models, was obtained from human-like trajectories and trajectories with different route leading to the merging point. This is not the case when predicting the heading, since an incorrect heading measurement, will lead to an unsuccessful trajectory recommendation.

5) Predicted vs Human Trajectories: The comparison of human trajectories against the predicted trajectories are vital when assessing the overall performance of the RL model. This is due to the fact that, the connected vehicle should match a human merge such there is minimal disruption on the road, nor cause any unexpected behaviour from unconnected vehicles unaware of the merge. Hence, if the RL model can achieve human like trajectories, then its feasibility as a solution for the lane merge coordination scenario significantly increases.

The agent has managed to mimic the recorded human-like trajectories for the successful merging instances, such case is presented in Fig. 13a. This is an interesting finding, since the reward function can be used to directly influence the agent behaviour to match a human trajectory, in order to achieve maximum intractability of vehicles on the road. However, another merges attempt to reach the target merge location via a different route, since the agent does not perceive the lanes surrounding the vehicles, it is hard for the model to determine the optimal path to undertake with respect to the lanes, as Fig. 13b shows. Nonetheless, this was a design choice in the RL model (see Section IV) to ensure maximum generalizability of the approach to different scenarios rather than, using a supervised learning approach to learn one scenario. Thus, the implementation and the architecture can be widely used across different scenarios with out the need to retrain the models.

B. Automotive and Communication KPIs for real vehicles

The lane merge scenario used for the real tests consists of a test track as described in Fig. 14 using connected and unconnected vehicles and the Lane Merge Coordination presented Fig. 1. For the *Image Recognition* system [24] was used and for the *GDM*, the *V2X Gateway* and the 5*G Network* with edge cloud capabilities the work from [26] was also used. Four vehicles were used, three are connected: merging, following and preceding vehicle, while the fourth vehicle was unconnected. This enables the trajectory recommendation to be passed to the merging vehicle for execution, whilst giving the TO some control over other connected vehicles in order to suggest a cooperative lane merging scenario benefiting the entire road. The TO is aware of unconnected vehicles due to the RUDs sent by the *Image Recognition* system.

In order to compare live TO's predicted trajectories and human trajectories, a preliminary test was implemented with no TO's interaction: several merges were performed on the test track while the KPI evaluation platform was storing the logs of those merges that occurred on the road. These stored logs are the human merges that are used to compare the predicted trajectory accuracy and the human likeness of the manoeuvre.

(a) Eagle-view of lane merge (b) Close-Up of lane merge

Fig. 14. Real lane merging scenario in test track using connected vehicles [6].

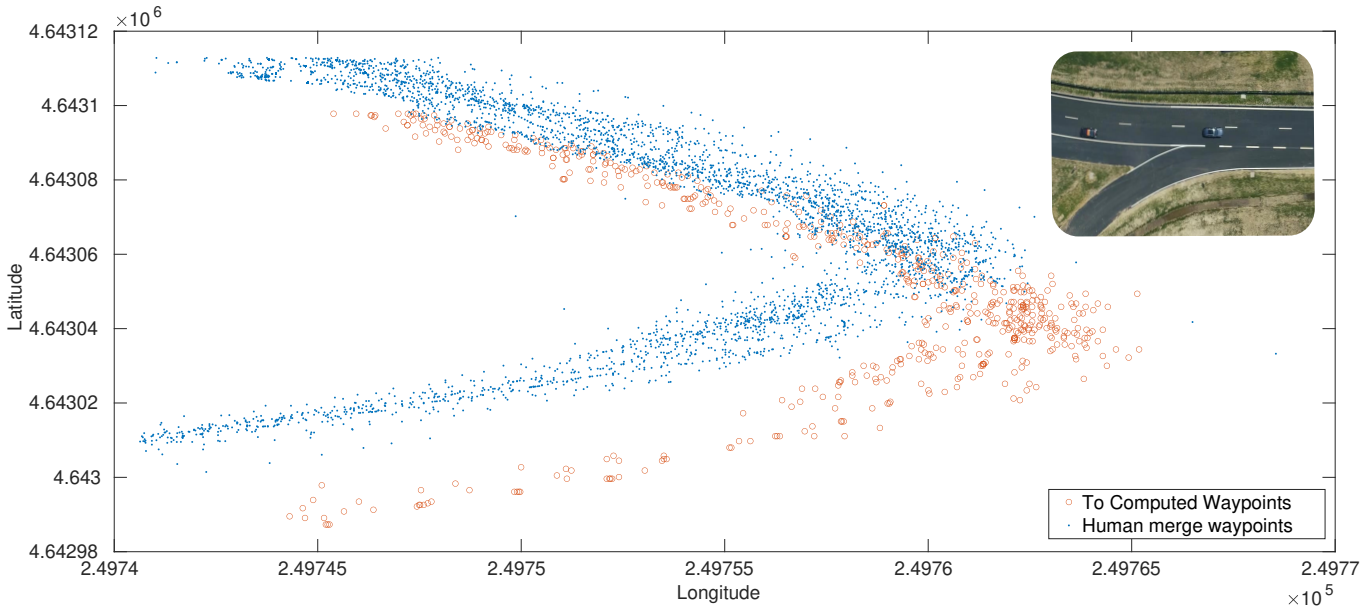
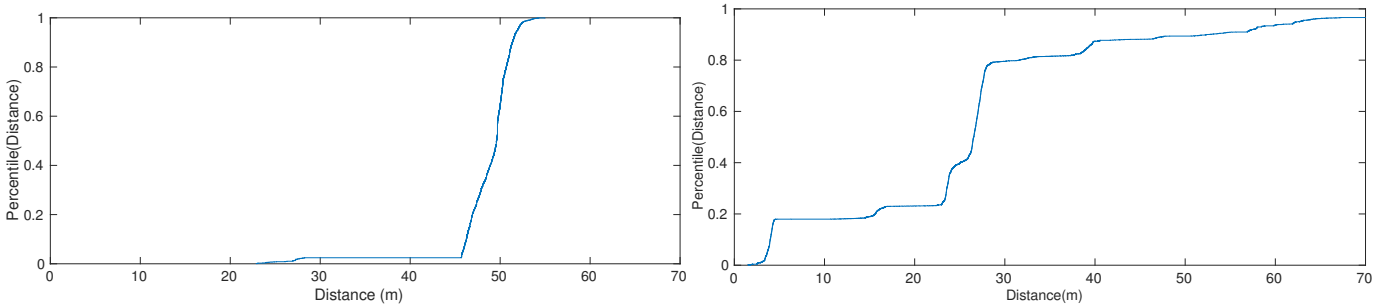


Fig. 15. Human vs computed trajectories based on their location on test track.

1) Predicted vs Human Collected Trajectory Positioning: Fig. 15 shows the latitude and longitude of the merging scenario for human and computed way points. The general shape of the merge has been detected and successfully predicted by the TO corresponding to the road architecture, this is a good indicator that the RL can adapt and generalise to real world scenarios that it has never encountered before. However, there is an obvious bias from the predicted way points. Since, the road information was removed from the training of the model to ensure a greater generalisation, has actually hindered the neural network to exactly follow a trajectory that a human may undertake. The *RL* also required perfect synchronisation of the environment in real-time, therefore, high frequency, low latency and great precision was required to ensure that the *TO* could feed the correct RUD to the *RL*. As such, the bias could stem from the minor delays the architecture incurred. The precision and accuracy of the architecture incorporating $5G$ and edge cloud was higher than the average [35], this could have not been obtained by using out-of-the-box implementation of the $5G$ communication spectrum. Therefore, there is a trade off between its ability to generalise the problem to the intended behaviour that is expected to achieve, with respect to the communication architecture the model is placed in.

2) Inter-Vehicular Distance: The inter-vehicle distances provide an insight on the merge of the connected vehicles in between the preceding and following vehicle. This value is mainly affected by the data fusion of the *Image Recognition* system and the actual connected vehicles broadcasting position. Fig. 16 presents the Empirical Cumulative Distribution Function (ECDF) of distance values recorded between vehicles laying on the same lane for the TO and human manoeuvres. On one hand, the largest amount of cases lies between $48 - 60m$ for Fig. $16a$. On the other hand, Fig. $16b$ shows that inter-vehicular distance varies greatly when the marge is undergone by humans spanning from $5 - 70m$. This means that human merges were performed under risky situations in some cases. In counterpart, the merging car does not hinder the safety distances between the other vehicles that are presented on the road, when calculated by the TO. In this sense, the TO does not bias the merging distance between the preceding and following car, opting to merge approximately in between the two vehicles, to maintain the largest inter vehicle distance



(a) TO calculated inter-vehicle distance

(b) Human calculated inter-vehicle distance

Fig. 16. ECDF of inter-vehicle distance during merging scenario.

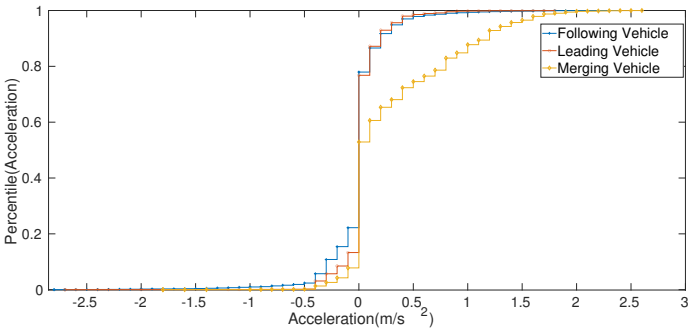


Fig. 17. ECDF of acceleration values during merging scenario.

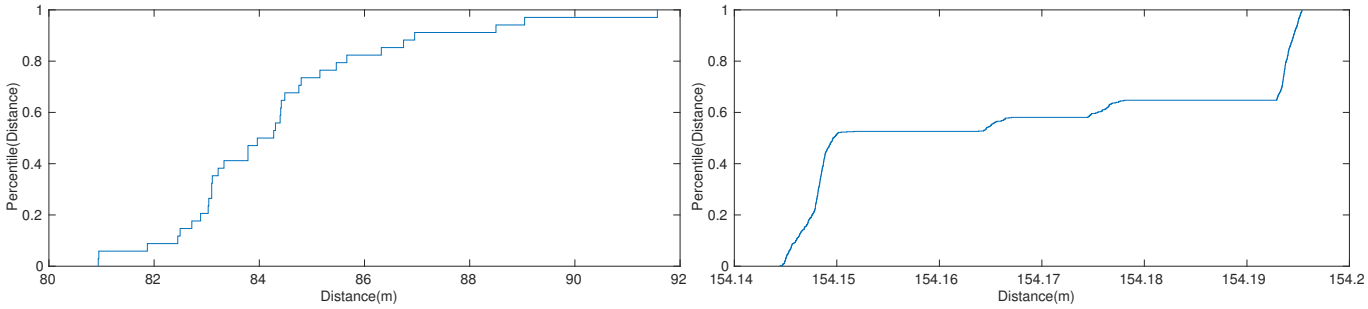
between the merging car and the two vehicles on the target lane. Although this does not reflect human-like driving in most cases, since the merging car can favour merging towards the preceding vehicle to allow more room for subsequent actions such as breaking, this was another design choice, in order to ensure the adaptability of the model and the approach to different lane merging scenarios, but also simplify the expected behaviour of the merging vehicle, to reduce neural network complexity and resources.

3) Merging Acceleration: Fig.17 presents the ECDF of acceleration values against the acceleration obtained in the scenario. The acceleration values given to the merging vehicle were concentrated on speeding the vehicle up to merge in between the two vehicles from a slow lane into the target merging lane. The majority of the acceleration values lied in the range $0 - 2\ m^2/s$ providing non extreme acceleration values for a merge mimicking a human driver approach to a lane merge. This means that the TO suggested acceleration values that provides a smooth trajectory recommendation during the merge. From the following vehicle's point of view, the recommendations given had the intended purpose of slowing down the following vehicle to create a larger gap in between the vehicles on the target lane, for a safer and smoother merge experience. Consolidating the idea of a coordinated lane merge approach taken on the road. The values obtained from the *V2X Gateway* displayed minor noise which further affected the speed and acceleration values recommended by the TO.

4) Manoeuvre Length: This distance is important when analysing the adaptability of the TO generated recommendations. This is based on the fact that most motorway planning procedures are designed with a certain length of merge that corresponds to an expected velocity and acceleration for a vehicle. In this case, the test track spanned $180m$ for the merging lane scenario. Fig. 18 depicts the count of distances obtained against the manoeuvre length. As seen from Fig. $18a$ the manoeuvre length remained in the range $81m$ to $91m$ solely focusing around the $80m$ mark for human manoeuvre, varying greatly between the range. On the other hand, the TO predicted manoeuvres stayed constant in the $154m$ region as seen in Fig. $18b$. This proves that the TO recommendations although not being trained with road parameters maintained a constant manoeuvre length corresponding to a successful merge without compromising safety procedures and adhering to the road structure. This ensures maximum adaptability of said recommendations in a merging scenario in a constrained road space scenario.

5) Trajectory Delivery Time: The vehicle *trajectory delivery time* was measured by the KPI Evaluation Platform using the methodology described in section V. Software components, mobile network and transport protocol have been identified as main contributors of the overall trajectory rate of delivery. However, the main propose of this KPI is to provide a network performance metric to be use as a baseline and we leave for future works an in-depth analysis of specific component delay and their possible improvements. Therefore, we focus on the *trajectory delivery time* per vehicle due to the time in-synchronicity among vehicles, software and network components. Fig. 19 shows $380,000$ measurements taken during the merging tests, where the count of the time taken to request and receive a trajectory recommendation is plotted against the rate obtained. We ca see that in 30% of the cases a *trajectory delivery time* of $50ms$ or less was obtained, also 99.9 percent of the measurements are under a rate of $288ms$ (receiving the locality of the vehicle or sending the trajectory recommendation is roughly $144ms$). Fig. 20 describes the standard deviation for the same measurements, where the mean value sits on $21ms$ (around $42ms$ for the entire communication path).

In terms of processing delays, the TO manoeuvre computation is negligible in the scenario. In most of the cases, it was not possible to obtain TO's recommendation computation time estimations due to the logging time granularity, which is a positive result. The TO achieved a real-time environment



(a) Human Manoeuvre (b) Predicted Manoeuvre

Fig. 18. ECDF of Manoeuvre Length.

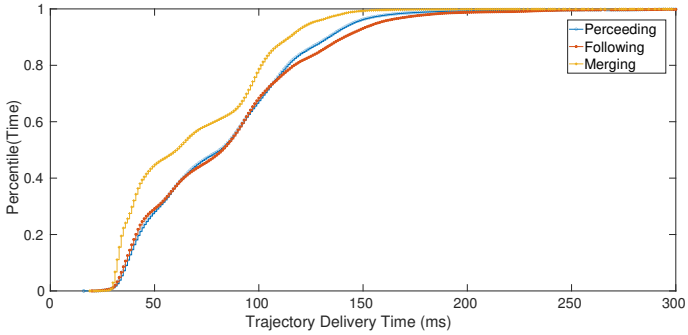


Fig. 19. ECDF of *trajectory delivery time* per vehicle.

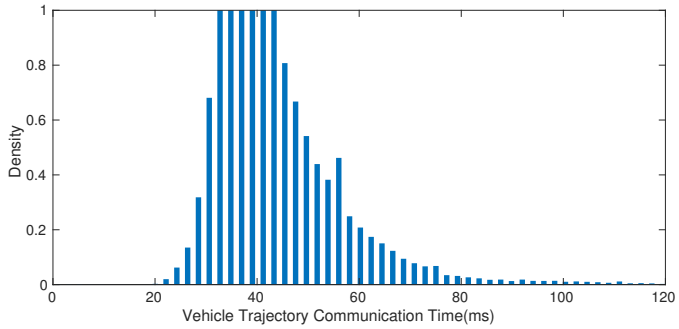


Fig. 20. Standard deviation of vehicle trajectory delivery time

processing, generating safe and successful manoeuvres for vehicles in need. On the other hand, the DF was identified as the component adding the larger computational delay. This could be due to the use of default TCP configuration, i.e, Nagle algorithm and Delayed Acknowledgements. Nagle algorithm improves the efficiency of TCP by reducing the number of packets that need to be sent over the network by delaying some small packets and sending them all at once. Similarly, Delayed Acknowledgement is used to reduce the number of acknowledgements sent back. The combination of the Nagle algorithm and the Delayed Acknowledgements could add up to $200ms$ to certain packets [36], which will negatively impact the added delay of the DF.

VII. Conclusion

In this paper, we presented a lane merge coordination model based on a centralised system. It delivers trajectory recommendations to connected vehicles on the road. Real tests were performed using a combination of connected and unconnected vehicles on a test track.

The *Traffic Orchestrator* and the *Data Fusion* components were implemented and tested, presenting meaningful results. The Dueling DQN model has been identified as the best approach compared to the DQN, obtaining more optimal performance and providing more human-like trajectories. Predicted trajectories provided smooth driving experience during the lane merge with mean acceleration in the range of $0-2\ m^2/s$.

Future works need to be carried out in order to improve *Data Fusion's* performance, in particular processing time and transport protocol optimisation are points to be addressed. Also, a hybrid implementation of a centralised and decentralised system could be considered to encompass the lane merging scenario. In this scenario, negotiation, communication and cooperation could enhance the dependability whereby catering to vehicles that may fall out of the network range.

Acknowledgment

This work has been performed in the framework of the H2020 project 5GCAR co-funded by the EU. The views expressed are those of the authors and do not necessarily represent the project. The consortium is not liable for any use that may be made of any of the information contained therein. This work is also partially funded by the EPSRC INITIATE EP/P003974/1 and The UK Programmable Fixed and Mobile Internet Infrastructure.

References

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 22–32, 2014.

[2] M. Dalla Cia, F. Mason, D. Peron, F. Chiariotti, M. Polese, T. Mahmoodi, M. Zorzi, and A. Zanella, "Using smart city data in 5g self-organizing networks," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 645–654, 2018.

[3] 5GAA, "The case for cellular v2x for safety and cooperative driving," November 2016. [Online]. Available: http://5gaa.org/wp-content/uploads/2017/10/5GAA-whitepaper-23-Nov-2016.pdf

[4] 5GPPP, "5g automotive vision," October 2015. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2014/02/5G-PPP-White-Paper-on-Automotive-Vertical-Sectors.pdf

[5] K. Antonakoglou, N. Brahmi, T. Abbas, A. E. Fernandez Barciela, M. Boban, K. Cordes, M. Fallgren, L. Gallo, A. Kousaridas, Z. Li, T. Mahmoodi, E. Strm, W. Sun, T. Svensson, G. Vivier, and J. Alonso-Zarate, "On the needs and requirements arising from connected and automated driving," *Journal of Sensor and Actuator Networks*, vol. 9, no. 2, 2020. [Online]. Available: https://www.mdpi.com/2224-2708/9/2/24

[6] 5GCAR, "5g communication automotive research and innovation," June 2019. [Online]. Available: https://5gcar.eu/

[7] D. Bevly, X. Cao, M. Gordon, G. Ozbilgin, D. Kari, B. Nelson, J. Woodruff, M. Barth, C. Murray, A. Kurt, K. Redmill, and U. Ozguner, "Lane change and merge maneuvers for connected and automated vehicles: A survey," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 105–120, 2016.

[8] H. C. Hsu and A. Liu, "Kinematic design for platoon-lane-change maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 9, no. 1, pp. 185–190, 2008.

[9] Q. H. Do, H. Tehrani, S. Mita, M. Egawa, K. Muto, and K. Yoneda, "Human drivers based active-passive model for automated lane change," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 42–56, 2017.

[10] J. Nilsson, J. Silvlin, M. Brannstrom, E. Coelingh, and J. Fredriksson, "If, when, and how to perform lane change maneuvers on highways," *IEEE Intelligent Transportation Systems Magazine*, vol. 8, no. 4, pp. 68–78, 2016.

[11] M. Zhu, J. Hu, L. Kong, R. Shen, W. Shu, and M. Wu, "An algorithm of lane change using two-lane nasch model in traffic networks," in *2013 International Conference on Connected Vehicles and Expo (ICCVE)*, 2013, pp. 241–246.

[12] J. Mar and Hung-Ta Lin, "The car-following and lane-changing collision prevention system based on the cascaded fuzzy inference system," *IEEE Transactions on Vehicular Technology*, vol. 54, no. 3, pp. 910–924, 2005.

[13] J. Suh, H. Chae, and K. Yi, "Stochastic model-predictive control for lane change decision of automated driving vehicles," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 6, pp. 4771–4782, 2018.

[14] S. Sivaraman, M. M. Trivedi, M. Tippelhofer, and T. Shannon, "Merge recommendations for driver assistance: A cross-modal, cost-sensitive approach," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, 2013, pp. 411–416.

[15] S. Bansal, A. Cosgun, A. Nakhaei, and K. Fujimura, "Collaborative planning for mixed-autonomy lane merging," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 4449–4455.

[16] S. Sivaraman and M. M. Trivedi, "Dynamic probabilistic drivability maps for lane change and merge driver assistance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 5, pp. 2063–2073, 2014.

[17] P. Wang and C. Chan, "Formulation of deep reinforcement learning architecture toward autonomous driving for on-ramp merge," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–6.

[18] A. R. Fayjie, S. Hossain, D. Oualid, and D. Lee, "Driverless car: Autonomous driving using deep reinforcement learning in urban environment," in *2018 15th International Conference on Ubiquitous Robots (UR)*, 2018, pp. 896–901.

[19] Q. Rao and J. Frtunikj, "Deep learning for self-driving cars: Chances and challenges," in *2018 IEEE/ACM 1st International Workshop on Software Engineering for AI in Autonomous Systems (SEFAIAS)*, 2018, pp. 35–38.

[20] V. Franois-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, *An Introduction to Deep Reinforcement Learning*, 2018.

[21] Y. Hou, P. Edara, and C. Sun, "Modeling mandatory lane changing using bayes classifier and decision trees," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 2, pp. 647–655, 2014.

[22] D. E. Kirk, "Optimal control theory: An introduction," -, 1970.

[23] H. Sahni. (2018) Reinforcement learning never worked, and 'deep' only helped a bit. [Online]. Available: https://himanshusahni.github.io/2018/02/23/reinforcement-learning-never-worked.html

[24] K. Cordes and H. Broszio, "Constrained multi camera calibration for lane merge observation," in *14th International Conference on Computer Vision Theory and Applications (VISAPP)*, February 2019.

[25] T. Bray, "Rfc 7159: The javascript object notation (json) data interchange format," *Internet Engineering Task Force (IETF)*, March 2014.

[26] M. Fallgren, T. Abbas, S. Allio, J. Alonso-Zarate, G. Fodor, L. Gallo, A. Kousaridas, Y. Li, Z. Li, Z. Li, J. Luo, T. Mahmoodi, T. Svensson, and G. Vivier, "Multicast and broadcast enablers for high-performing cellular v2x systems," *IEEE Transactions on Broadcasting*, vol. 65, no. 2, pp. 454–463, 2019.

[27] L. Sequeira, A. Szefer, J. Slome, and T. Mahmoodi, "A lane merge coordination model for a v2x scenario," in *2019 European Conference on Networks and Communications (EuCNC)*, 2019, pp. 198–203.

[28] Z. Wu, N. M. Khan, L. Gao, and L. Guan, "Deep reinforcement learning with parameterized action space for object detection," in *2018 IEEE International Symposium on Multimedia (ISM)*, 2018, pp. 101–104.

[29] A. Notsu, K. Yasuda, S. Ubukata, and K. Honda, "Optimization of learning cycles in online reinforcement learning systems," in *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2018, pp. 3530–3534.

[30] S. Mukhopadhyay, O. Tilak, and S. Chakrabarti, "Reinforcement learning algorithms for uncertain, dynamic, zero-sum games," in *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2018, pp. 48–54.

[31] A. Jeerige, D. Bein, and A. Verma, "Comparison of deep reinforcement learning approaches for intelligent game playing," in *2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC)*, 2019, pp. 0366–0371.

[32] A. D. Pambudi, T. Agustinah, and R. Effendi, "Reinforcement point and fuzzy input design of fuzzy q-learning for mobile robot navigation system," in *2019 International Conference of Artificial Intelligence and Information Technology (ICAIIT)*, 2019, pp. 186–191.

[33] M. Wang, L. Wang, and T. Yue, "An application of continuous deep reinforcement learning approach to pursuit-evasion differential game," in *2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, 2019, pp. 1150–1156.

[34] K. Janocha and W. M. Czarnecki, "On loss functions for deep neural networks in classification," 2017.

[35] X. Ge, "Ultra-reliable low-latency communications in autonomous vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 5, pp. 5005–5016, 2019.

[36] K. Shin, J. Kim, K. Sohn, C. Park, and S. Choi, "Online gaming traffic generator for reproducing gamer behavior," in *Entertainment Computing - ICEC 2010*, H. S. Yang, R. Malaka, J. Hoshino, and J. H. Han, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 160–170.