Towards an Optimal MEC Resources Dimensioning for Vehicle Collision Avoidance System: A Deep Learning Approach

Bouziane Brik and Adlen Ksentini
EURECOM, Sophia Antipolis, France
Email: name.surname@eurecom.fr

***Abstract*—Collision detection and avoidance between vehicles is one of the key services envisioned in the Internet of Vehicles (IoV). Such services are usually deployed at the Multi-access Edge Computing (MEC) to ensure low latency communication and thus guarantee real-time reactions to avoid collisions between vehicles. In order to maximize the coverage of the road and ensure that all vehicles are connected to an optimal MEC host (in terms of geographical location), the collision avoidance application needs to be instantiated on all the MEC hosts. This may add a burden on the computing resources available at the latter. In this paper, we propose an AI-empowered framework that aims to optimize the computing resources at the MEC hosts. Our framework uses deep learning to (1) predict the vehicle density to be served by a MEC host and (2) derive the exact computing resources required by the collision detection application to run optimally. We evaluate the proposed framework using a real dataset representing vehicle mobility in a big city. Obtained results show the accuracy of our prediction model and hence, the efficiency of our resources assignment framework to exactly deduce the optimal computing resources needed by each instance of the application.**

***Index Terms*—Internet of Vehicles, Collision Avoidance, MEC, Deep Learning, LSTM, 5G**

I. Introduction

In the vision of 5G and the Internet of Vehicles (IoV), connected vehicles and autonomous driving are among the key envisioned applications; they impact not only mobile operators as well as car manufacturers business, but also our everyday life [1]. Connected vehicles and autonomous driving involve several components, such as sensors, actuators, and applications, which need to coordinate in order to achieve the envisioned autonomy of vehicles. Among the critical services towards autonomous driving is the collision detection/avoidance system. It consists of continuously collecting data from vehicles and using these data to predict collisions and communicate alerts or send commands to vehicles in order to avoid collisions with other vehicles. The collision detection/avoidance service comprises (1) an application that runs at the vehicles and collects data, such as GPS coordinates, speed, acceleration, and (2) a remote application hosted at the cloud infrastructure that runs a collision detection algorithm. The latter may send control commands to the vehicles, when deemed appropriate, such as reduce speed, change direction, or break. One of the main requirements to run the collision avoidance service, in 5G and beyond, is to dispose of a low-latency connection between the client and server-side of the application, i.e., between the vehicles and the remote application sitting in the infrastructure. To ensure low-latency communication, it is envisioned to deploy several instances of the collision application. For instance, at the road intersections and close to base stations using Multi-access Edge Computing (MEC) [2]. This ensures that each vehicle is connected to the closed application instance, which guarantees low latency communication. To recall, MEC consists of deploying computation capability (hosts) close to the end-users, for instance, at the vicinity of base stations [3]. Thus, all data can be treated locally without involving the remote cloud server, hence reducing latency and the traffic to carry throughout the network. MEC hosts are distributed all over the network, constituting a distributed and low-latency computation resource for delay-sensitive applications, like vehicle collision detection/avoidance applications. Indeed, the latter need low latency communication with the remote vehicles since the control commands, such as break or reduce speed, need to be received by the vehicles in near real-time to react to any threat and avoid collisions. As mentioned earlier, one pertinent solution is to locate several instances of the collision detection application at the MEC, hence reducing the end-to-end communication latency. The different instances can be used to cover all the road intersections; each vehicle shall always be in contact (connected) with an instance of the application. However, duplicating the number of instances of the application may add a burden on the MEC host computing resources, such as CPU usage, memory, storage, etc. It is worth noting that MEC host resources are limited compared to central cloud servers. In this context, it is vital for MEC operators (generally the network operator) to optimize the MEC resource usage, particularly when considering that 5G will rely, among others, on MEC to support services that require low-latency such as data caching, Virtual and Augmented Reality (VR/AR), etc. Hence, ensuring an efficient share of the MEC computing resources is critical.

In this paper, we propose a novel framework that relies on Deep Learning (DL) to predict vehicles' mobility and accordingly assign computing resources to the vehicle collision detection application instances, aiming at better optimizing the overall MEC resources and ensuring optimal functioning of the service (i.e., guaranteeing low response time). The proposed framework predicts using DL, and more precisely Long Short-

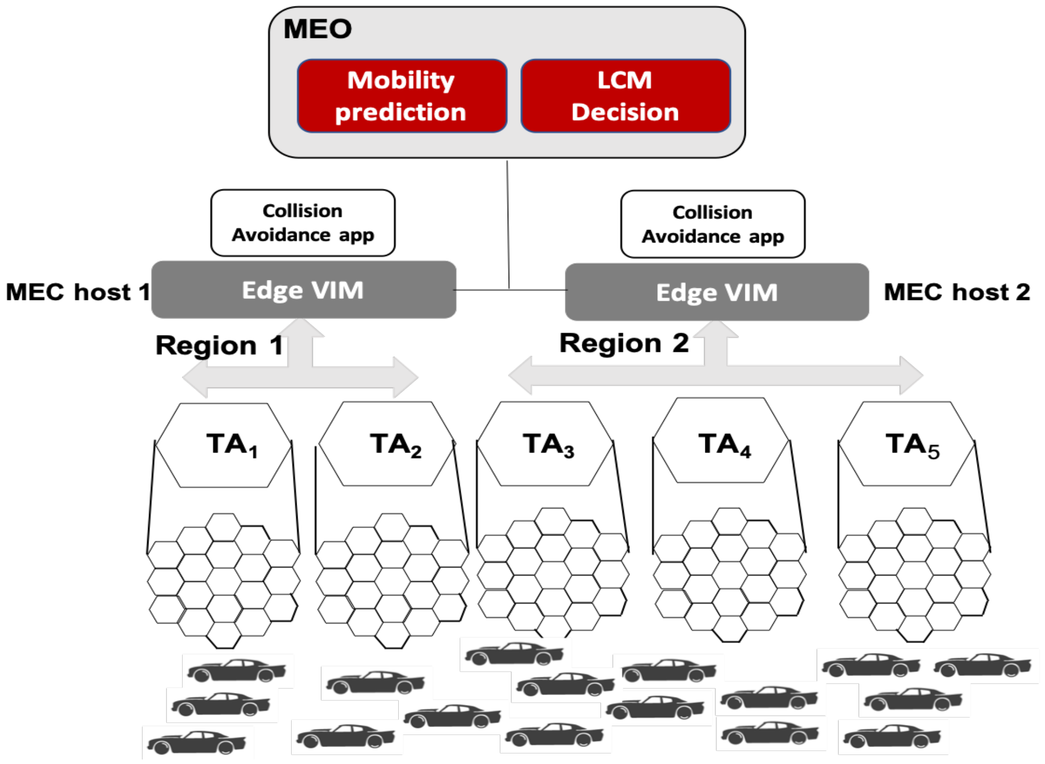

Fig. 1. The Envisioned Architecture and Deployment.

Term Memory (LSTM), the mobility of the vehicles, and according to their positions in the network, it derives the needed MEC resources. On one hand, increase MEC resources for the collision avoidance application instances that serve a high number of vehicles; on the other hand, reduce these resources for the instances that cover a low number of connected vehicles. The contributions of this paper are as follows:

- Define an AI-empowered framework that relies on ETSI MEC to optimally deploy the vehicle collision detection and avoidance service.
- Train and build a DL module based on LSTM, using a real dataset to predict the vehicles' mobility.
- Devise a resource assignment algorithm that runs at the MEC orchestrator (MEO) and considers the mobility prediction of vehicles when assigning computing resources to the running application instances.

The rest of the paper is organized as follows. Section II provides state of the art on the usage of MEC to support vehicles collision detection and avoidance systems, whereas Section III describes our proposed framework in terms of mobility prediction model as well as resource assignment algorithm. Section IV is devoted to the performance evaluation of the proposed framework evaluation and the obtained results analysis. Finally, conclusions are drawn in Section V.

II. Related work

Employing the MEC system to optimize the collision detection and avoidance service in IoV has been rarely explored in the literature. Only a few works exist. These works usually start by collecting vehicles' mobility information in real-time before detecting collisions and alerting the concerned vehicles. In [4], the authors address one of the relevant classes of automotive services, called the Extended Virtual Sensing (EVS) services, which among others, shows the benefit of using edge computing resources in the context of autonomous driving. Particularly, the authors focus on the EVS application that supports vehicle collision avoidance at intersections before describing its implementation using the OpenAirInterface MEC platform [5]. The EVS application identifies possible collisions by performing in real-time the distances between vehicles.

An enhanced Collision Avoidance (eCA) service is proposed in [6]. eCA is deployed at the edge of the network and comprises mainly two algorithms: (i) Collision Avoidance Algorithm (CAA), which first determines the vehicle's next trajectory via the positional information included in the periodic beacons and then detects if two vehicles are on a collision path or not. In particular, the next vehicles' trajectories are predicted by projecting the next vehicles' positions onto curve or straight segments. This projection depends on the status of the blinking lights; (ii) Collision Avoidance Strategy (CAS), which notifies the vehicles potentially involved in a collision regarding the action needed to avoid it.

In [7], the collision avoidance application for vehicular networks is extended to benefit vulnerable users, e.g., pedestrians and bicycles, equipped with a smartphone. The authors proposed a MEC-based collision avoidance system. Through Basic Safety Message (BSM), this system periodically collects users' information such as position, acceleration, speed, and heading, in order to estimate users' trajectories and avoid collision between them.

Although the above works addressed the collision avoidance issue between vehicles through the MEC system, they did not consider the MEC resources usage given the limited capacity of MEC hosts compared to the centralized cloud servers. Another drawback we can mention is the collision detection scheme that depends mainly on the performance of the collision avoidance system. In fact, these works are based on simple schemes to estimate the next trajectories of mobile users, for instance, by only determining the distances between vehicles in real time [4], or reading the status of the blinking lights [6]; while in this paper, we explore the usage of DL to predict vehicles mobility and use this prediction to improve the management of MEC computing resources.

III. Proposed Framework

A. Architecture

As stated earlier, we assume that the collision avoidance application is duplicated (i.e., several instances) and deployed on all the MEC hosts of a network operator, allowing to cover, with low-latency access, a large geographical area. We recall that a MEC host includes a virtualization platform that runs applications' instances in the form of Virtual Machines (VM) or Containers. Each MEC host has a computing capacity depending on the used hardware, which is limited compared to a centralized cloud. Fig. 1 depicts the envisioned architecture. We assume that a MEC host is deployed to cover a specific geographical location, which corresponds to a set of gNBs (5G Base station) organized in a Tracking Area (TA). A TA is a concept used in cellular networks, which consists of grouping together a set of cells. The aim is to optimize the mobility management algorithm by simplifying the procedures inside a TA group. In Fig. 1, MEC host1 covers a geographical location composed of TA1 and TA2. All connected vehicles moving in this area are served by the collision avoidance application instantiated at MEC host1. If a vehicle moves from TA2 to TA3, then it will be served by the collision avoidance application instantiated in MEC host2. It is important to note that the redirection of the traffic from collision avoidance application instance 1 to instance 2 is transparent to the vehicles. Indeed, to ensure a seamless redirection of the traffic to the new instance, we rely on Domain Name Service (DNS). The DNS servers record the IP addresses of the running application instances along with their geographical location. Therefore, when a vehicle tries to connect to the MEC application, it has to resolve the service's URL to an IP address. In this case, the DNS server will send the IP address of the closest MEC application instance. To avoid the DNS cache's impact at the vehicle, we use a small value of the Time To Leave (TTL) of a cache entry, leading to often the resolution of URL of service.

One crucial challenge to successfully deploy collision avoidance applications in the context of IoV is to ensure broad coverage of the network with low latency connectivity. This can be achieved in 5G by using MEC and duplicating the application on all the MEC hosts to guarantee a broad coverage of the network, hence the road. Obviously, as the number of application instance increases, the consumed overall MEC resources increases. In this work, we assume that one instance of the application can efficiently handle, without increasing the response time, N connected vehicles. Indeed, the application's response time is a critical metric to ensure low end-to-end latency. In the context of a virtualized environment in MEC, the collision application instance will run as a MEC application (MEapp) on top of a virtualization platform as a Container or a VM. Therefore, a MEapp instance will use a certain number of computing resources, namely virtual CPU (vCPU), which should be optimal to ensure a low response time of the running service. We assume that a MEapp consuming X vCPU can optimally (keep a low response time) handle N users. To handle $2N$, then $2X$ CPUs are needed. Later we will describe how we can obtain these values.

To recall in MEC, the MEC Orchestrator (MEO) manages the Life Cycle Management (LCM) and orchestrates the computing resources of the MEapps. It is in charge of deploying the MEapps on top of the Virtualized platform at the MEC host. The MEapps are described using an Application Descriptor (AppD), which includes configuration information, such as the application image, and the computing resources needed by the application. The MEO then is in charge of requesting the CPU resources and updating the request if deemed appropriate by scaling up or down the needed CPU, i.e., increase or decrease the number of vCPU assigned to a MEapp. For more details on ETSI MEC architecture, interfaces, and components, readers may refer to [8].

In this work, our objective is to derive at the MEO, for each MEapp instance, the needed number of vCPU, by finding a trade-off between optimizing the MEC host computing resources and ensuring low response time to optimally run the service. Intuitively, one solution would be that each time a vehicle (or a batch of vehicles) has moved from one MEC host to another MEC host, the resource management algorithm runs at the MEO computes the needed vCPU for each application instance it manages. However, this solution requires continuous tracking of vehicles' mobility at the network layer, which is difficult to enforce. Accordingly, we propose to leverage the above-mentioned solution with a mobility prediction model using LSTM to anticipate the update of resources (vCPU) needed by each MEapp instance. The aim is to not only optimize the MEC resources, but also ensure that a MEapp instance performs optimally (low response time) considering the number of vehicles connected to it.

The proposed algorithm runs at the MEO, as shown in Fig. 1. It is composed of two modules, the first one, namely the mobility prediction module, which takes as inputs the vehicle GPS coordinates obtained from the collision avoidance MEapps (via Mp1 interface [8]) and predicts through the LSTM module, the next position of the vehicles; while the second module, namely LCM decision, uses the predicted next vehicle positions to derive the needed computing resources (vCPU) of each MEapp instance, which allow to optimally run the collision avoidance service. The MEO enforces the LCM decisions via the MEC Edge Platform Manager (MEPM), which according to ETSI, is in charge of updating the CPUs resources of the running MEapps through the MEC host/Edge Virtualization Infrastructure Manager (MEP/VIM) [8]. The global algorithm runs as follows:

- Initialise the configuration of all MEapp instances of the collision avoidance application with X_0 CPUs.
- **Epoch Loop**
    1) Mobility prediction: Receive a batch of GPS coordinate from the MEapp
    2) Mobility prediction: Predict the new location of vehicles for epoch $t+1$, and hence the number of users to be connected for each MEapp instance.
    3) LCM: run a decision algorithm that (1) takes as inputs the predicted number of served vehicle per MEapp for $t+1$, (2) extracts the necessary computing resources according to the predicted number, and (3) changes the configuration of the MEapp instances, through the MEPM, if deemed appropriate.

In the next sections, we will detail the mobility and resources prediction as well as LCM modules.

B. Vehicles mobility prediction model

To predict the vehicles' mobility, we use deep Recurrent Neural Networks (RNN) with Long-Short-Term-Memory (LSTM) algorithm. The main procedures are: the dataset, designing the neuronal network, training the neuronal network, and testing the network. The three first steps are described in the following sub-sections, while the last step is described in the performance evaluation section.

1) Taxis Mobility Data: We use real and publicly available taxi trace data, which is composed of 464019 records and gathered over 30 days in San Francisco (USA) [9]. This dataset was collected in May 2008 and contains mobility traces in terms of GPS coordinates of approximately 500 taxis. Each taxi is equipped with a GPS module and sends periodically, each 10 sec, its location (Timestamp, ID, Geo-coordinates) to a central server.
It is worth noting that we consider the San Francisco map as (n x m) grid cells, and we translate each taxi's GPS coordinates to a cell ID. The dataset is used to provide actual taxis' GPS coordinates to our mobility prediction model in order to predict the next ones, and hence the vehicles' next location, which is used as input by the resource assignment algorithm to update or not the assigned computing resources to the running MEapp instances. We use this dataset rather than simulated GPS coordinates to validate our algorithm under real mobility traces.

2) Design of Taxis Mobility Prediction Model: RNN with LSTM algorithm is well-suited to classify, process, and predict time series, given time lags of unknown duration [10]. In fact, RNN with LSTM algorithm is capable of learning long-term dependencies between input data by using an internal memory to remember past data in memory. This makes it suitable for our problem to predict the next location of vehicles (Cell IDs) based on the past one.

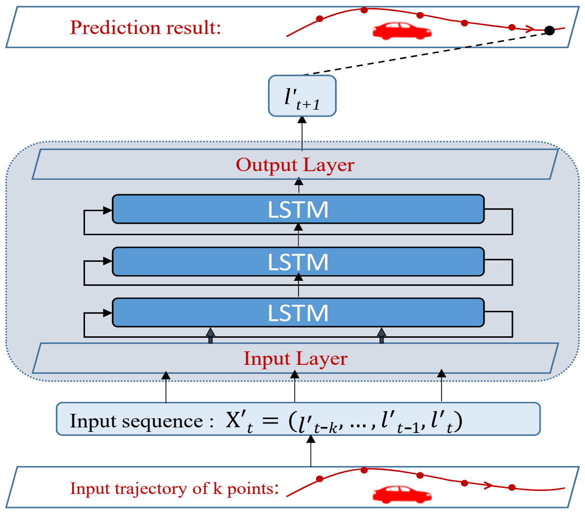
Prediction result:
l'_{t+1}
Output Layer
LSTM
LSTM
LSTM
Input Layer
Input sequence : $X'_t = (l'_{t-k}, \ldots, l'_{t-1}, l'_t)$
Input trajectory of k points:

Fig. 2. Basic LSTM-Based prediction Model of vehicles location.

Fig. 2 illustrates our LSTM-based vehicles mobility prediction model. The prediction process comprises three main steps. The input vehicle's trajectory is first processed by a fully connected input layer with 56 neurons where each vehicle's position (Cell ID) is mapped to a 56-dimensional feature tensor. Then, the resulted sequence is sent to a deep RNN composed of three stacked LSTM layers, each with 56 neurons.
Each LSTM layer considers the previous LSTM layer's output as input and feeds its output to the next LSTM layer. Finally, a fully connected output layer with 45 neurons maps the output of the last LSTM layer to the cell ID, corresponding to the predicted vehicle's cell ID of the next time-step, l_{t+1}. We argue the usage of 45 neurons by the fact that we divided the San Francisco map into 45 cells,
Finally, the training of the model phase aims at minimizing the distance between the predicted and real location of vehicles (Cell ID). Hence, we choose the Mean Squared Error (MSE) as the loss function and adopt the Stochastic Gradient Descent algorithm to update the neural network parameters [11].

C. LCM Resources Assignment

As indicated earlier, the LCM module runs the decision algorithm that may request resources update for the running instances of the collision avoidance MEapp. Let assume $v()$, $loc()$, $u()$ as vectors that respectively represent the vCPU used by a MEapp, the location of a MEapp (MEC server ID), and the number of vehicles connected to a MEapp. The index of the vector corresponds to the collision avoidance MEapp instance number. We note by v_t, u_t, the value of the vectors at epoch t. At the initial epoch ($t = 0$) we

note $v\_0 = \{c_0, c_0, \ldots, c_0\}$, and $u\_0 = \{0, .., 0\}$, where c_0 corresponds to the initial configuration of the MEapp (i.e. number of vCPU), u_0 is the initial number of connected vehicles to the collision avoidance application. The decision algorithm is detailed in Algorithm 1, where N is the number of instances, $Change()$ is a vector of Boolean, and $C(X)$ is a function that gives the necessary vCPU to optimally handle X users. $C(X)$ is an integer value between 1 and M. Note that C(X) can be derived by benchmarking an instance of the collision avoidance application, which can be obtained by simulation or using a real deployment. The C(X) function aims to indicate the number of users that can be handled by one instance while ensuring the computing latency very low, hence reducing the response time.
Firstly, the decision algorithm considers as input the predicted number of vehicles to be served by each MEapp instance for the next epoch. Secondly, it verifies if the current number of vCPU used by the MEapp is not optimal (more resources are needed or over usage of the resources). If so, then an update of the resources is requested. Finally, for all the MEapp instances that need an update, a request is sent to the MEPM.

IV. Performance evaluation

We divide the performance analysis of the proposed framework into two parts: (1) the performance and accuracy of the mobility prediction module, and (2) the LCM resource assignment performances. For the first part, we used the Tensorflow engine to implement our RNN with the LSTM-based prediction model [12]. In addition, we compared our learning model with two other learning algorithms: (i) RNN with Gated Recurrent Unit (GRU) based model, which is similar to LSTM, but has fewer parameters than LSTM, as it lacks an output gate [13]. (ii) Convolutional Neural Network (CNN) based model, a deep learning algorithm designed to process arrays of data, and can also be applied for time series forecasting problems [14]. Table I presents the considered parameters to compare the three algorithms. For the sake of fairness in terms of comparison, the three algorithms share the same activation function and optimizer (Stochastic Gradient Descent).
For the second part, we simulated a distributed MEC system that covers the city of San Francisco. The city map has been divided into (n x m) grid cells. We assume that the grid cells are covered by six MEC hosts, and each one (host) is in charge of covering a set of cells. Besides, we assume that each collision avoidance instance needs one vCPU to manage two vehicles. It should be noted that this number can be fixed by benchmarking the application in real deployment or by simulation. Finally, we suppose that each MEC host has 15 vCPU, which means that a MEapp instance may get at a maximum of 15 vCPU; thus, it can manage up to 30 vehicles. We then compare our resources assignment scheme to a static scheme that assigns a fixed number of vCPU (10 vCPU) to each MEapp instance, whatever the vehicles' density (i.e., the number of vehicles to be served by a MEapp instance). We focus on two main metrics to validate our scheme: overloaded MEapp instances (i.e., the demand exceeds servers capacity in terms of vCPU resources) and over-provisioned MEapp instances (i.e., less than 20% of MEapp's vCPU resources are used); which corresponds, respectively, to a high response time of the application instance, and an under optimal usage of the MEC host computing resources.

TABLE I
Implementation parameters.

Parameters	**Values**
Dataset	
Number of records	464019
Number of cars	500
Collection time period	May 2008
Percentage of training set	80% of the dataset
Percentage of test set	20% of the dataset
Deep Learning	
Deep learning Tool	Tensorflow
LSTM timestamp Window	60
Activation functions	Relu (hidden layers) Softmax (output layer)
Optimizer	Stochastic Gradient Descent (SGD)
Loss function	Mean Squared Error
Learning rate	0.01
Dropout	0.2
Batch size	50 samples
Number of epoch	$[20, 60]$ epochs
MEC	
Number of MEC Hosts	6 hosts
Number of vCPU	15 vCPU

A. Mobility Prediction Evaluation

Figs. 3-(A) compares the considered learning algorithms' performances in terms of Mean Squared Error (MSE) on the test dataset, aiming at validating the performances of our prediction model using unseen data, i.e., data that the models have not seen before. We remark that our LSTM-based model minimizes the MSE compared to the other algorithms even when we increase the number of test samples. We also observe that CNN and RNN with GRU generate almost the same performances. To validate these results, we depict in Fig. 3-(B) a comparison between the real and predicted cars' cell ID values of ten test samples. Mostly, we notice that the predicted cell IDs using the LSTM-based model are similar to the real cell ID values. However, RNN with GRU and CNN algorithms fail in predicting the correct cell ID for some test samples. For instance, test samples ID = 2, 3, 8 and 9. These results confirm Fig. 3-(A) results, i.e., the efficiency and the accuracy of the LSTM-based model in predicting cars' cell IDs.

B. Resource Assignment Evaluation

Fig. 4-(A) and Fig. 4-(B) represent, respectively, the vehicle density under the coverage of each MEC host (and hence the vehicles' number to be served by a MEapp instantiated at the MEC host) during a rush hour (from 8 am to 9 am), and the number of overloaded MEapp instances during that hour. The number of overloaded MEapp instances is a critical metric for the service-level performance as an overloaded MEapp

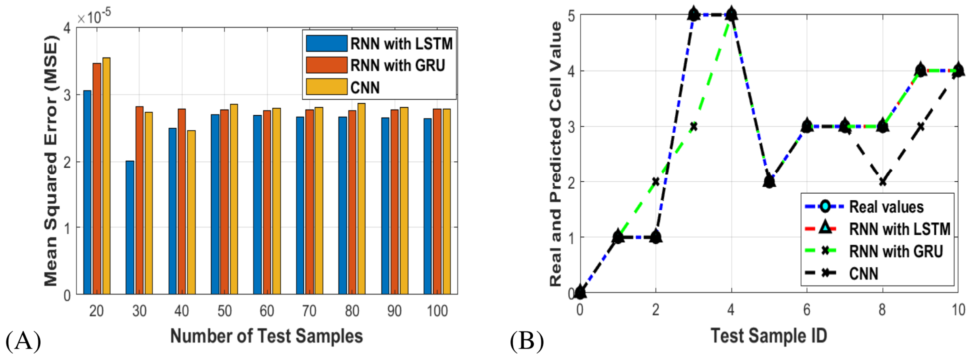

Fig. 3. Performances comparison between learning algorithms. (A) Mean Squared Error on test dataset. (B) Real and Predicted cell ID values.

means high response time, which degrades the performance and may lead to safety issues. From Fig. 4-(A), we observe that the vehicle density is high; most of the MEapp has more than 20 vehicles to serve, which requires more than 10 vCPU. We remark in Fig. 4-(B) that our scheme keeps the number of overloaded MEapp instances very low compared to the static scheme. We argue this by the fact that the LSTM-based model can predict the density of vehicles in each group of cells and hence can anticipate the needed vCPU resources of the collision avoidance instances (i.e., MEapp instances), which allows adapting to the vehicles' density to be served. However, by fixing the vCPU value to be used by each MEapp instance, the static scheme results in a high number of overloaded MEapp instances. This is mainly due to the fact that the demand (vehicle density) exceeds the MEapp instances' capacity (i.e., 20 vehicles), which can clearly be observed in Fig. 4-(A).

Fig. 5-(A) and Fig. 5-(B) illustrate, respectively, the vehicle density during the low traffic hours (from 3 pm to 4 pm) and the number of over-provisioned MEapp instances during that hour. From Fig. 5-(A), we observe that the vehicle density is low, and most of the MEC host covers less than 10 vehicles, which requires only 5 vCPU. We notice from Fig. 5-(B) that again, our scheme minimizes the number of over-provisioned MEapp instances compared to the static scheme. In fact, anticipating the required resources also helps to reduce the number of over-provisioned MEapp instances, as the LCM computes exactly the needed number of vCPU to use with the current vehicle density to serve. However, in the static scheme, several MEapps are over-provisioned. Only 5 over 10 vCPU are used to manage the current vehicle density, which leaves 5 unused and cannot be assigned to another MEapp instance in the MEC host.
To summarize, we can deduce that our resources assignment algorithm's performance depends mainly on the accuracy of the LSTM-based prediction model. The generated results show the efficiency of our LSTM-based model to estimate the number of vehicles at each cell, which in turn improves the efficiency of our resources assignment algorithm in forecasting exactly the needed resources by each MEapp instance; hence improving the overall resource of the MEC host. Contrariwise, the static scheme uses the same computing resources, whatever the density of vehicles to serve for all the MEapp instances, which leads to degrade both the MEC computing resource usage and the service performance.

V. Conclusion

In this work, we introduced an AI-empowered framework that aimed to improve the management of MEC resources (mainly computing) when deploying vehicle collision detection and avoidance service in IoV. The framework aimed at finding a trade-off between improving the overall usage of MEC resources and guaranteeing that each instance of the deployed vehicle collision detection and avoidance application is assigned enough resources to optimally run, i.e., low response time. To find this trade-off, our framework uses RNN with LSTM to predict vehicles' density at each cell and then computes the exactly needed computing resources by each instance of the application.

We built our RNN with LSTM-based model and evaluated our framework using a real dataset of vehicle mobility in a big city. The obtained results showed the accuracy of our prediction model to estimate the exact number of vehicles at each group of cells and the efficiency of the resource allocation resource to optimize both the overall MEC resources and the application performance.

One of our future work directions is to perform positions prediction in a distributed way, using distributed learning, which will allow the vehicle to make the prediction locally.

Acknowledgement

This work has been partially supported by the European Union's H2020 MonB5G (grant no. 871780) project and the ANR Chist-era Leading Edge project.

References

[1] H. Cao, S. Gangakhedkar, A. R. Ali, M. Gharba, and J. Eichinger, "A 5g v2x testbed for cooperative automated driving," in *2016 IEEE Vehicular Networking Conference (VNC)*, 2016, pp. 1–4.

[2] H. Ma, S. Li, E. Zhang, Z. Lv, J. Hu, and X. Wei, "Cooperative autonomous driving oriented mec-aided 5g-v2x: Prototype system design, field tests and ai-based optimization tools," *IEEE Access*, vol. 8, pp. 54 288–54 302, 2020.

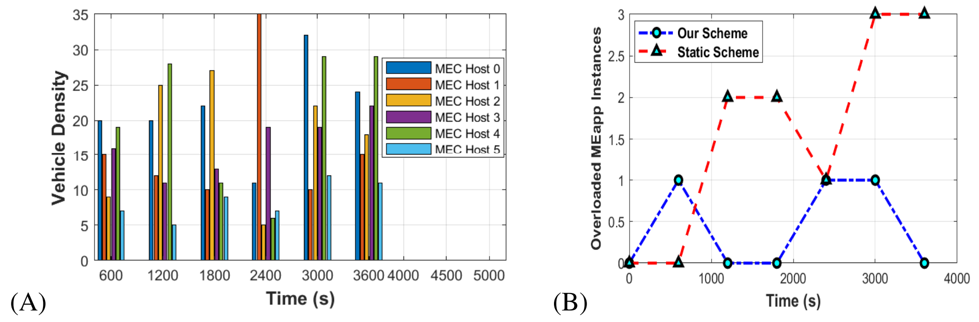

Fig. 4. Performance Evaluation of our Resources Assignment Algorithm (8am-9am). (A) Vehicle Density per MEC Host. (B) Overloaded MEapp instances.

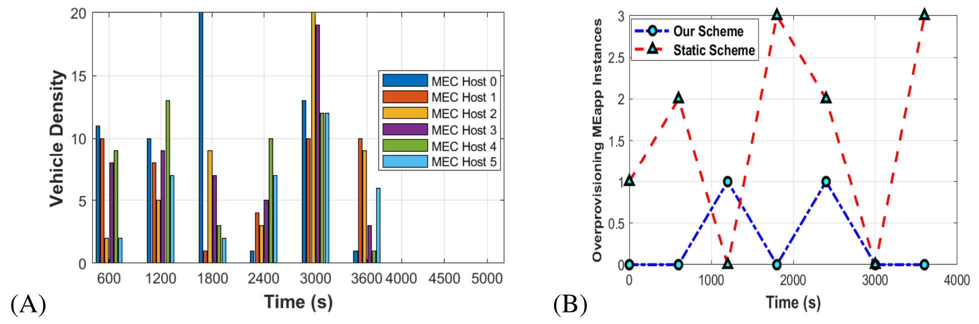

Fig. 5. Performance Evaluation of our Resources Assignment Algorithm (3pm-4pm). (A) Vehicle Density per MEC Host. (B) Over-provisioned MEapp instances

[3] A. Huang, N. Nikaein, T. Stenbock, A. Ksentini, and C. Bonnet, "Low latency MEC framework for sdn-based LTE/LTE-A networks," in *IEEE International Conference on Communications, ICC 2017, Paris, France, May 21-25, 2017*. IEEE, 2017, pp. 1–6.

[4] G. Avino, P. Bande, P. A. Frangoudis, C. Vitale, C. Casetti, C. F. Chiasserini, K. Gebru, A. Ksentini, and G. Zennaro, "A mec-based extended virtual sensing for automotive services," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1450–1463, 2019.

[5] S. Arora, P. A. Frangoudis, and A. Ksentini, "Exposing radio network information in a mec-in-nfv environment: the rnisaas concept," in *5th IEEE Conference on Network Softwarization, NetSoft 2019, Paris, France, June 24-28, 2019*, C. Jacquenet, F. D. Turck, P. Chemouil, F. Esposito, O. Festor, W. Cerroni, and S. Secci, Eds. IEEE, 2019, pp. 306–310.

[6] M. Malinverno, J. Mangues-Bafalluy, C. E. Casetti, C. F. Chiasserini, M. Requena-Esteso, and J. Baranda, "An edge-based framework for enhanced road safety of connected cars," *IEEE Access*, vol. 8, pp. 58 018–58 031, 2020.

[7] M. Malinverno, G. Avino, C. Casetti, C. F. Chiasserini, F. Malandrino, and S. Scarpina, "Mec-based collision avoidance for vehicles and vulnerable users," 2019.

[8] A. Ksentini and P. Frangoudis, "Toward slicing-enabled multi-access edge computing in 5g," *IEEE Network*, vol. 34, no. 1, pp. 99–105, January 2020.

[9] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "CRAWDAD dataset epfl/mobility (v. 2009-02-24)," Downloaded from https://crawdad.org/epfl/mobility/20090224, Feb. 2009.

[10] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, p. 1735–1780, Nov. 1997.

[11] C. De Sa, M. Feldman, C. Ré, and K. Olukotun, "Understanding and optimizing asynchronous low-precision stochastic gradient descent," in *2017 ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, 2017, pp. 561–574.

[12] P. Goldsborough, "A tour of tensorflow," 2016.

[13] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014.

[14] B. Zhao, H. Lu, S. Chen, J. Liu, and D. Wu, "Convolutional neural networks for time series classification," *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, pp. 162–169, 2017.

Biographies

BOUZIANE BRIK received his Ph.D degree from Laghouat and La Rochelle (France) universities in 2017. He is currently working as associate professor at Burgundy (Bourgogne) university and DRIVE laboratory. He has been working on network slicing in the context of H2020 European project on 5G. His research interests include also Internet of Things (IoT), IoT in industrial systems, Smart grid, and vehicular networks.

ADLEN KSENTINI is an IEEE COMSOC distinguished lecturer. He obtained his Ph.D. degree in computer science from the University of Cergy-Pontoise in 2005. Since March 2016, he is a professor in the Communication Systems Department of EURECOM. He has been working on several EU projects on 5G, Network Slicing, and MEC.