AutoMEC: LSTM-based User Mobility Prediction for Service Management in Distributed MEC Resources

Umberto Fattore*
Marco Liebsch
{name.surname}@neclab.eu
NEC Laboratories Europe GmbH
Heidelberg, Germany

Bouziane Brik
Adlen Ksentini
{name.surname}@eurecom.fr
Eurecom
Biot, France

ABSTRACT

The 5^{th} generation of the cellular mobile communication system (5G) is in the meantime stepwise being deployed in mobile carriers' infrastructure. Various standardization tracks as well as research activity are investigating the exploitation of the very flexible 5G system architecture for customized deployments, meeting requirements of the vertical industry, such as for automotive, factory, or smart city. A very common base is a cloud-native development and decentralized deployment of the 5G system along with services in distributed resources per the Multi-Access Edge Computing (MEC) architecture to locate services topologically close to (mobile) users, e.g. along public roads, and to enable low-latency communication with local services. Automated management of such a distributed deployment in an agile environment is a prerequisite. This paper investigates the use of Recurrent Neural Networks (RNN) for accurate user mobility prediction in an automotive scenario. By the use of simulated vehicular traffic, a suitable RNN configuration using Long Short-Term Memory (LSTM) has been found, which provides accurate prediction results. Proof of value has been accomplished by an experimental decision algorithm, which balances the use of available distributed resources through service scale, migration or replication decisions while meeting mobile users' expectation on the experienced service quality.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; • **Networks** → **Mobile networks**; Network simulations; *Cloud computing*.

ACM Reference Format:
Umberto Fattore, Marco Liebsch, Bouziane Brik, and Adlen Ksentini. 2020. AutoMEC: LSTM-based User Mobility Prediction for Service Management in Distributed MEC Resources. In *23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM '20), November 16–20, 2020, Alicante, Spain.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3416010.3423246

*Umberto Fattore is also PhD student at Universidad Carlos III de Madrid

MSWiM '20, November 16–20, 2020, Alicante, Spain

ACM ISBN 978-1-4503-8117-8/20/11…$15.00
https://doi.org/10.1145/3416010.3423246

1 INTRODUCTION

A first stable release of the cellular mobile communication system's 5^{th} generation is currently being built and stepwise deployed. Its design follows a clean separation of control- and data plane functions, which can be flexibly deployed in a centralized or distributed manner. Various research projects as well as standardization tracks investigate a cloud-native as well as a customized deployment of the 5G system in support of the vertical industry's demand. As example, the European Commission is funding three cross-corridor projects, such as the 5G-CARMEN project [12], which investigate deployment options and extensions to the 5G system for the automotive industry towards connected, cooperative and automated mobility (CCAM). In order to meet associated communication requirements of vehicles, a distributed deployment of the 5G system's data plane functions along with distributed automotive service functions topologically close to the cars enables short communication paths between a service and a car as its consumer, which can result in low-latency communication.

A cloud-based and distributed deployment of compute, networking and storage resources, which can instantiate and offer virtualized network- and service functions close to mobile users, is denoted as Multi-access Edge Computing (MEC) and is standardized in the ETSI ISG MEC [3]. Research activities go far beyond MEC deployment in large and powerful data centers, but consider highly distributed, though resources- and energy constrained MEC platforms, such as lightweight and low-power micro-data centers or even system-on-chip (SoC) solutions for deployment along roads or even on drones [4]. Such distributed MEC deployment requires thorough management of scarce resources. Furthermore, in order to enable service continuity as well as low-latency services in such a highly dynamic environment, a mobile user may need to be relocated from a service running on a currently used MEC platform to another MEC platform and service to keep topological distance and associated communication latency low. Continuous adoption to such agile environment requires automated monitoring, assessment and re-configuration of connectivity as well as the virtualized network- and service deployment.

This paper investigates the use of Recurrent Neural Networks (RNNs) for accurate user mobility prediction in an automotive scenario to allow proper management of distributed MEC resources. From various configurations, a best match in terms of RNN complexity, learning phase convergence and prediction accuracy is determined. Network services are treated as virtualized service instances, which can be deployed on each MEC platform, consuming defined resources and serving a limited number of users. Adding or

removing service instances (scaling) or moving instances from one MEC platform to another one (migration) in order to maintain low-latency and keep a service 'local' to the user can be aligned with predicted number of users in a particular geographic area, which is served by a dedicated MEC platform, in order to optimize the utilized distributed MEC resources. An experimental decision algorithm is used to proof the value of accurate mobility prediction for service deployment runtime decisions, achieving a best match between users' expected service quality (i.e., low-latency) and optimized utilization of MEC resources.

2 RELATED WORK

Multi-Access Edge Computing (MEC) [3] is a strongly emerging approach allowing for offloading of latency-critical services to MEC platforms located in proximity of users, therefore potentially achieving lower latency and traffic offloading. Whereas details on the positioning and number of MEC platforms adopted strongly varies based on the specific deployment [10], leveraging such MEC locations by being able to constantly allocate latency-critical services near to the final user offers a strong occasion for powering future mobile networks [11]. This results also in the necessity to find a trade-off between both the available constrained resources of MEC locations (e.g., storage, computational power) and strict requirements demands from mobile users. Taking service allocation decisions among different MEC platforms therefore would benefit from the capability to anticipate the future needs of the network and in particular of mobile users populating it [9]. By doing this, services can be allocated in MEC locations proactively. Assuming the scope is to keep the topological distance between the service and the user short, the main enabler for a proactive management of services in a distributed MEC environment is given by user mobility prediction.

Many works focus on user mobility prediction, for instance predicting the presence of a specific mobile user in a Point of Interest (PoI), e.g., Home, Work. [5] does this extending Mobility Markov Chains (MMC), [13] uses Neural Networks (NNs), while [8] Spatial-Temporal extended Recurrent Neural Networks (RNNs) applied on a dataset obtained through Gowalla.[1] All the aforementioned works achieve good accuracy but request for an high amount of historical information per user. Other solutions use a grid-based approach, in which the user location is bound to a specific cell of a grid (e.g., mapped to the nearest base station): both [15] and [14] use RNN Long Short-Term Memory (LSTM) to perform user mobility prediction, the latter in order to allow dual-connectivity on base stations.

The cited works focus on the prediction of mobility for each user: therefore, this results in the need of predicting singularly the future movements of each user in order to have an overall picture of user density for each location. In a dense-populated distributed MEC scenario (i.e., thousands of mobile users moving between different MEC locations), this approach is highly complex and poorly scalable. Whereas some solution taking in account specifically optimization for application relocation in MEC (i.e., using deep reinforcement learning) exists, it is based on a waypoint mobility model in which users are simply assumed to move constantly between two randomly generated points [2]. In our solution, we focus as well on a prediction oriented towards the need of service management in MEC, but considering the density distribution of users in a grid-based environment rather than a single user mobility pattern; more, we then consider a real map with urban streets and highway in order to make a more proper evaluation.

[1] users voluntarily "check-in" at different geographical locations when visiting them.

3 RNN-BASED AUTOMATED MANAGEMENT OF DISTRIBUTED MEC RESOURCES

In order to proof the value of an accurate prediction, we utilize an experimental decision algorithm to decide when allocating or removing instances of a service on a MEC location (i.e., a geographical region served by dedicated MEC resources). The defined algorithm takes into account service requirements, MEC-related parameters and in particular the expected number of users for each location. To properly obtain the latter, it is important to have a strong prediction method which can assure a significantly accurate prediction of the number of users per location. Based on this information, the decision algorithm is used to take any of the following decisions for each location : (1) **Migration**: move instances from neighbor locations to the current location (i.e., migrating instances of the service), (2) **Replication**: replicate a service instances on the current location without removing an instance on neighbor locations, (3) **Scale**: add/remove one or more service instances at the current location, or (4) **Retain**: keep the current allocation as it is.

3.1 Prediction Algorithm

The objective of the prediction algorithm is to predict the future number of users in a specific location (i.e, grid cell), given historical data on the density of users in those locations in the previous time samples. This can be summarized as a time series prediction, for which a typical solution is given by *Recurrent Neural Network (RNN)*. RNNs are able to detect patterns in sequential information, by learning the relationship between present inputs and previous ones: this is done using hidden layers and a recursively hidden state (h_t), which is at each time reused as input for the next time computation, therefore keeping memory of the sequence history.

Basic RNNs however suffer a well-known problem: most recent inputs in time are strongly prioritized while old inputs are easily forgotten, resulting in a kind of "short memory" behavior. This can be avoided by using a special *Long Short-Term Memory (LSTM)* unit for the RNN [6], which enforces a set of operations carrying information of previous inputs. In particular, this is done using an additional state called *cell state* (c_t), which is at each step partially or totally forgotten based on its importance and then updated with the new input, through the use of *sigmoid* and *tanh* functions respectively. The output hidden state (h_{t+1}) takes into account the effect of the cell state. An alternative to LSTM is given by *Gated Recurrent Unit (GRU)*, which offers a simplification to LSTM complexity by implementing a similar behavior but with a reduced set of operations and without the need of the additional cell state [1]. Whereas for some particular cases [1] GRU could offer similar accuracy as LSTM but with a reduced complexity, this is not always true and depends on the specific configuration and dataset utilized. In this paper we compare LSTM and GRU in terms of learning time and prediction accuracy. The configuration with the best match has been selected for deployment and evaluation with the experimental decision algorithm.

3.2 An Experimental Decision Algorithm for the Management of Distributed Services

The objective of the experimental decision algorithm is to minimize the amount of overall utilized resources while meeting users' service delay bounds. We assume that none, one or multiple instances of a service are allocated in a cell and each instance can serve a limited number of users (capacity bounds). A finite amount of available resources for service instances (resource bounds) is available at each MEC location. A user will always be served by the nearest available service instance relative to its position, which can be either at its current MEC location, resulting in lower communication delay, or in a neighbor location, resulting in saving resources at its current MEC location. If there is an instance with sufficient capacity allocated in the same cell of a user, he will be served by the co-located instance at the minimum latency. Whenever this will not be possible (e.g., no instance allocated in the cell or all instances have already reached the maximum number of users), the user will be served from a neighbor cell with higher latency.

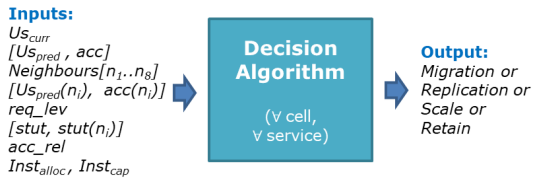

Figure 1: Decision Algorithm input and output parameters.

Fig.1 depicts input (i.e., parameters based on which a decision is made) and output (i.e., the decision outcome) of the algorithm, which runs for each cell and takes a decision whether or not to allocate one or multiple instances of the service. A *migration* consists in allocating an instance and removing it from a neighbour cell, whereas a *replication* consists in allocating a new instance without removing it from a neighbour cell. The latter case allows serving more users locally with reduced latency but at the cost of utilizing more resources. Increasing or reducing the number of instances is also a possibility (*scale*), as well as not changing anything (*retain*).

Input includes the current number of users for the cell Us_{curr}, and both the predicted number of users for the cell Us_{pred} and all neighbor cells $Us_{pred}(n_i)$, as well as the estimated accuracy for each prediction (*acc* and $acc(n_i)$) and a list of all *Neighbours* cells. An accuracy relevance *acc_rel* parameter is used to keep more or less in consideration the estimated accuracy, while a requirement level *req_lev* parameter is used for each service to indicate how strong the low-latency requirements are for that service in being deployed locally at a user's current MEC location (e.g., an higher value forces the decision towards the local allocation of instances despite the potential drawback in terms of resource consumption). Finally, resource-related parameters include the storage utilization for the cell and the neighbour cells (*stut*) and the number of instances already allocated for the service in the cell $Inst_{alloc}$ as well as the maximum capacity of instances $Inst_{cap}$.

The algorithm pseudo-code is depicted in Algorithm 1. There are two main conditions based on which a decision is made. In the first condition ($Us_{pred} > \mu^* Us_{current}$), the predicted number of users for the cell is compared with the number of users at the current

Algorithm 1: Experimental Decision Algorithm

```
Result: Migrate/Replicate/Scale/Retain
Us_pred = Run prediction algorithm for the cell
if Inst_allocated < Us_pred/Inst_capacity then
    μ = stut/req_lev + acc_rel/acc
    if Us_pred > μ * Us_current then
        N+ = (Us_current − μ*Us_pred)/Inst_cap
        Try allocating ⌈N+⌉ instances in current cell
        foreach n in Neighbors do
            ρ = req_lev/stut(n) + acc_rel/acc(n);
            if Us_pred > ρ * Us_pred(n) then
                N− = (Us_pred − ρ * Us_pred)/Inst_cap
                Remove⌊N−⌋instances in neighbor cell
```

time: a significant increase of the number of users predicted for the next time sample results in the need to allocate new instances of the service. This condition takes into account a weight factor μ, calculated based on parameters such as the storage utilization *stut* to allocate instances in the cell, the requirement level *req_lev* of a low-latency service (higher value means higher requirement for low-latency and therefore higher probability to be served by a local instance) and the accuracy of the prediction *acc*. If the local storage utilization is high, it is more likely that the decision will be to avoid a new allocation. In case the first condition has a positive value, the number of instances N_+ to be allocated is calculated based on the actual variation of users, the capacity of the existing instances $Inst_{cap}$ and the available storage. The second condition ($Us_{pred} > \rho^* Us_{pred}(n)$) is used to decide whether or not to remove one or multiples instances from a neighbor cell. This condition is checked for each neighbor cell. A comparison between the number of users predicted for the current cell Us_{pred} and the number of users predicted for neighbor cells $Us_{pred}(n)$ is made. Again, this condition takes in account a weight factor ρ considering similar parameters of the previous multiplier μ. For instance, a high storage utilization for the neighbor cell results in a low ρ value and therefore the decision algorithm will more likely remove instances of the service from the neighbor cell and keep it only on the current cell. If this happens, an instance migration has occurred and therefore all users connected to the service instance in the neighbor cell are now forced to use an instance of the service from a different cell (user migration). Similarly to the previous first condition, the number of instances to remove N_- is calculated and such number of instances are removed from the neighbor cell.

4 EVALUATION AND RESULTS ANALYSIS

As a prerequisite for the targeted evaluation, we generated realistic vehicles' positions data using Sumo simulation tool [7] and a real map in Germany including urban and roads and a highway. Then we evaluate different RNN configurations (i.e., LSTM and GRU) with different settings (i.e., number of layers) to identify the best configuration for our type of data. Finally, we use the predicted data to feed the decision algorithm and evaluate the gain in terms of service delay and storage consumption with the predicted data. Details of each phase are described in the following sections.

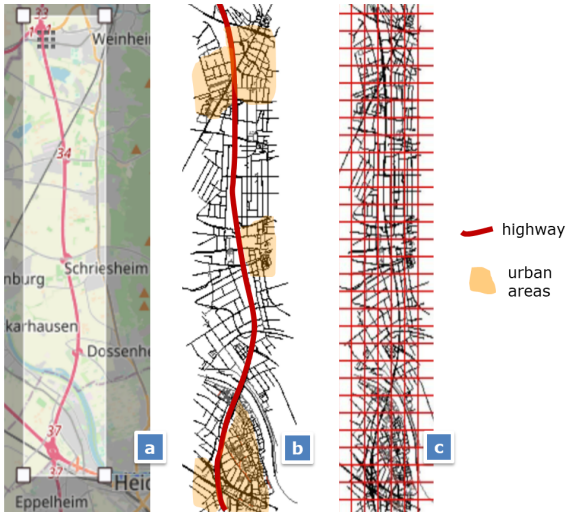

Figure 2: Real map used (a), street map imported in Sumo (b) and grid cell division of the map (c).

4.1 Creating vehicle mobility data with SUMO

The map used, depicted in Fig. 2, has been selected through OpenStreetMap[2] tool, and comprises a 20 km segment of highway A5 in Germany, between Heidelberg and Weinheim cities, including three exits and three entrances on both side of the highway and three correspondent small urban areas (less than 500 nodes each).

Vehicles are injected using SUMO simulation tool [7], for a total duration of 24 hours, in which three different families of flows are injected: (1) *Highway traffic*, comprising flows of vehicles on the highway (or entering the highway at a random entrance in the map) and driving all along the highway (or exiting at a random exit in the map), generated at various rates during the different periods of the day; (2) *Urban traffic*, comprising vehicles moving along 12 different routes between different points of the same urban area, generated only during daylight hours at a rate of 1 vehicle every 10 seconds; (3) *Background random traffic*, comprising additional vehicle flows moving using a totally random pattern, generated during the entire simulation at a 1 vehicle every 30 seconds rate. Speed of vehicles is defined with a normal distribution having mean value as 0.8 of the max street speed limit (e.g., 50 m/s on the highway) and std.dev. equal to 0.2 of said speed limit.

The map is then divided into 500x500 meters squared cells, each representing a MEC location, and for each cell the vehicle density, every time sample of 10 second, is considered. Fig. 2 depicts the original map portion selected, the node graph of streets obtained once imported in SUMO and the division in cells of the map.

4.2 Selection and tuning of RNN configuration

In order to choose the best configuration for our generated data set, we train and test in TensorFlow[3] our RNN network comparing results between LSTM and GRU configurations with 2, 3 or 4 layers (with 50 units for each layer). In particular, we consider a 2 hour interval (i.e., 16016 samples in total), collected from all cells

[2]online tool - https://www.openstreetmap.org/
[3]tensorflow: https://www.tensorflow.org/

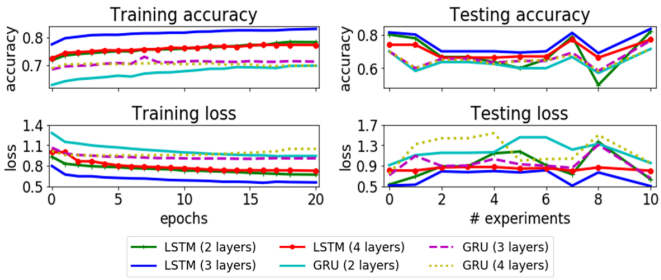

Figure 3: Accuracy and loss for training and test in RNN, using LSTM and GRU with different number of layers.

	2 layers	3 layers	4 layers
LSTM	3341s	3847s	5347s
GRU	3434s	5344s	14536s

Table 1: RNN training time (s)

in the map and aggregated in chunks of 20 samples each: therefore, 20 previous samples are used to predict the next (21st) sample. The number of epochs is fixed to 20, dropout value to 0.2 and optimizer and output layer activation function respectively to 'adam' and 'softmax' configuration.

Figure 3 shows both training accuracy and loss for each epoch, as well as accuracy and loss in the testing phase, the latter based on experiments made on 10 different data sets. Results show in general better performances achieved through LSTM: in particular, lower training accuracy values are achieved using GRU with 2 layers (around 0.68 after more than 15 epochs), while the highest accuracy is achieved using LSTM with 3 layers (above 0.80 after 3 epochs and around 0.85 after 10 epochs). In line with this, loss is lower in the case of LSTM configurations.

Finally, Tab. 1 depicts the time in seconds needed to train each of the different networks. Once more, LSTM seems to outperform GRU with in average a lower time (similar when considering 2 layers, about -30% for 3 layers and more than 120% less for 4 layers). Such results were obtained with a 32Gbyte RAM HP Aspire laptop.

4.3 Prediction evaluation

In order to evaluate the prediction performed on a distributed MEC environment, we feed the experimental decision algorithm with predicted data and we evaluate: (1) the *overall delay* for all cells at each time step, calculated as the average delay of every vehicle connected to the service, assuming said delay to be equal to *d*, *2d* or *3d* based on if the vehicle is served respectively by a service instance located in its same cell, in a neighbor cells, or elsewhere; (2) the *percentage of storage available* for each time sample, calculated as the total of the available storage of all cells; and (3) an *Efficiency* value, defined as the ratio between the available storage and the overall delay for all cells at a given time sample, normalized between 0 and 1. A higher efficiency value represents an higher capability of finding an advantageous trade-off between keeping latency low and storage availability high.

We compare the decision algorithm, which is fed with LSTM-based predicted values, with two other configurations: in one configuration, the algorithm is fed with past historical data, hence the

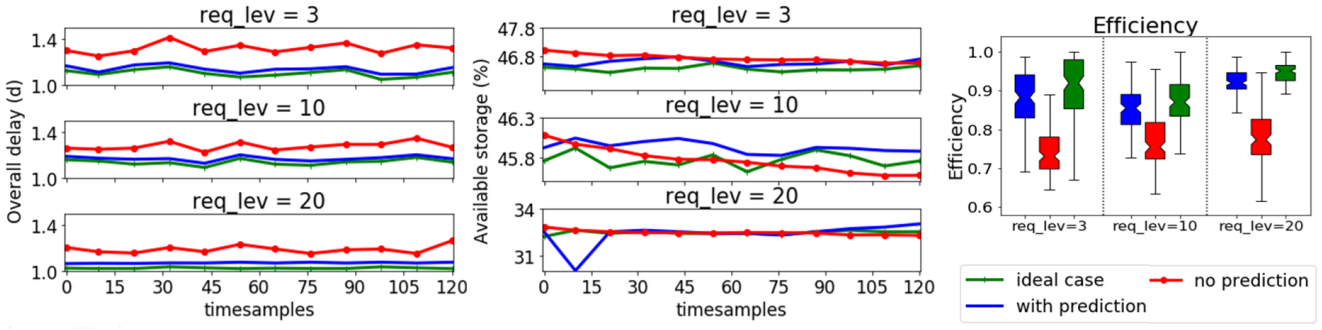

Figure 4: Decision algorithm overall delay, available storage and efficiency evaluation, against different req_lev values.

two main conditions of the algorithm are based on the variation of number of users at each location in past time samples (no prediction). This allows a comparison with the algorithm without leveraging proactive service management. In a further configuration we evaluate the ideal case in which the predicted number of vehicles is 100% accurate, as we feed the algorithm with real future data available from the simulation data set. This configuration determines the upper bound KPIs for comparison.

Fig.4 shows results referred to the 3 KPIs described, against different values of *req_lev* (i.e., =3, =10 and =20), and considering a total of 120 time samples (i.e., 20 minutes is the overall simulation time). The initial available storage is generated randomly between 1 and 10, while each instance of a service always consumes 1 storage unit and is able to serve a maximum of 10 vehicles. In order to simplify the readability of the results, the graphs depicting overall delay and available storage consider average values every 10 samples. In general, it can be observer that a higher *req_lev* results in a lower delay for vehicles, as well as in a decrease of the available storage; this is due to the fact that the service is recognized to have higher low-latency requirements and therefore more instances are allocated (i.e., relaxing the resource constraints), in order to allow vehicles to be served by locally deployed instances. Fig. 4 also shows that the decision algorithm leveraging LSTM-based predictions offers a strong gain compared with the case in which past data variation are used. In particular, whereas the case without prediction shows to be a bit less aggressive in utilizing available storage (about 2% less), it results in significantly higher delay values, (about at least 10% more). Therefore, using data predicted through our RNN LSTM algorithm guarantees higher efficiency, meaning it is able to find a better trade-off between latency requirements and storage utilization, as can be seen in the efficiency graph still in Fig. 4. More, our algorithm using predicted data behaves very similar to the ideal case: all KPIs results to be very similar and the difference in terms of efficiency is in average less than 3%. This is due to the high accuracy of our predicted values.

5 CONCLUSIONS AND FUTURE WORK

In this work, we test different RNN LSTM and GRU configurations in order to make a significantly accurate prediction of mobile users density in a distributed MEC environment. We evaluate such predictions with an experimental decision algorithm, managing distributed services in order to find a trade-off between MEC-related parameters and service requirements. Whereas the results depicted show the effectiveness of the predictions performed, further steps may be done towards a more concrete deployment and an associated tailored and improved resources optimization algorithm.

ACKNOWLEDGMENTS

The research leading to these results has been partially supported by H2020-MSCA-ITN-2016 framework under grant agreement No. 722788 (SPOTLIGHT) and H2020-ICT-2018 under grant agreement No. 825012 (5G CARMEN).

REFERENCES

[1] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* (2014).
[2] Fabrizio De Vita, Giovanni Nardini, Antonio Virdis, Dario Bruneo, Antonio Puliafito, and Giovanni Stea. 2019. Using Deep Reinforcement Learning for Application Relocation in Multi-Access Edge Computing. *IEEE Communications Standards Magazine* 3, 3 (2019), 71–78.
[3] ETSI. 2019. *Multi-access Edge Computing (MEC); Framework and Reference Architecture.* TS. European Telecommunication Standards Institute (ETSI).
[4] Umberto Fattore, Marco Liebsch, and Carlos J. Bernardos. 2018. UPFlight: An enabler for Avionic MEC in a drone-extended 5G mobile network. *IEEE 23rd Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)* (2018).
[5] Sébastien Gambs, Marc-Olivier Killijian, and Miguel Núñez del Prado Cortez. 2012. Next place prediction using mobility markov chains. In *Proceedings of the first workshop on measurement, privacy, and mobility.* 1–6.
[6] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[7] Daniel Krajzewicz. 2010. Traffic simulation with SUMO–simulation of urban mobility. In *Fundamentals of traffic simulation.* Springer, 269–293.
[8] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. 2016. Predicting the next location: A recurrent model with spatial and temporal contexts. In *Thirtieth AAAI conference on artificial intelligence.*
[9] Stavros Ntalampiras and Marco Fiore. 2018. Forecasting mobile service demands for anticipatory MEC. In *2018 IEEE 19th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM).* IEEE, 14–19.
[10] Sonia Shahzadi, Muddesar Iqbal, Tasos Dagiuklas, and Zia Ul Qayyum. 2017. Multi-access edge computing: open issues, challenges and future perspectives. *Journal of Cloud Computing* 6, 1 (2017), 30.
[11] Tarik Taleb and Adlen Ksentini. 2013. Follow me cloud: interworking federated clouds and distributed mobile networks. *IEEE Network* 27, 5 (2013), 12–19.
[12] The 5G-CARMEN project. 2019. *5G CARMEN Use Cases and Requirements.* Deliverable D2.1. https://5gcarmen.eu/publications/
[13] Jan Tkačík and Pavel Kordík. 2016. Neural turing machine for sequential learning of human mobility patterns. In *2016 International Joint Conference on Neural Networks (IJCNN).* IEEE, 2790–2797.
[14] Chujie Wang, Zhifeng Zhao, Qi Sun, and Honggang Zhang. 2018. Deep learning-based intelligent dual connectivity for mobility management in dense network. In *2018 IEEE 88th Vehicular Technology Conference (VTC-Fall).* IEEE, 1–5.
[15] Dilranjan S Wickramasuriya, Calvin A Perumalla, Kemal Davaslioglu, and Richard D Gitlin. 2017. Base station prediction and proactive mobility management in virtual cells using recurrent neural networks. In *2017 IEEE 18th Wireless and Microwave Technology Conference (WAMICON).* IEEE, 1–6.