Towards Edge-Assisted Trajectory Prediction for Connected Autonomous Vehicles

Mehdi Salim Benhelal, Badii Jouaber, Hossam Afifi, Hassine Moungla

Towards Edge-Assisted Trajectory Prediction for Connected Autonomous Vehicles

Mehdi Salim Benhelal
SAMOVAR
Telecom SudParis, Institut Polytechnique de Paris
Palaiseau, France
salim.benhelal_mehdi@telecom-sudparis.eu

Badii Jouaber
SAMOVAR
Telecom SudParis, Institut Polytechnique de Paris
Palaiseau, France
badii.jouaber@telecom-sudparis.eu

Hossam Afifi
SAMOVAR
Telecom SudParis, Institut Polytechnique de Paris
hossam.afifi@telecom-sudparis.eu

Hassine Moungla
Laboratory of Informatics Paris(LIPADE)
Université Paris Cité
hassine.moungla@u-paris.fr

***Abstract*—Trajectory prediction has been identified as a challenging critical task for achieving full autonomy of the connected and autonomous vehicles (CAVs). Despite the advancement of communication technologies, only few studies include the connectivity and data exchange aspects. Thus, we introduce a novel Edge-Assisted clustering architecture that takes advantage of recent deep learning models and the evolution of edge technologies to achieve better forecasting. First, the historical positions of the target vehicles are fed into the base models of all CAVs in the scene, resulting in multiple generated predictions. Then, each prediction is transmitted to an edge server where trajectories clustering is performed using DBSCAN algorithm to obtain multiple partitions with similar trajectories. The largest cluster is averaged then broadcast back to all CAVs in the scene. Our proposed method surpasses state-of-the-art results on the real world trajectory prediction nuScenes vehicles dataset, obtaining better predictions up to 21%. We also demonstrate the robustness of our method against single-agent system failures, succeeding to get very satisfactory results due to our ability to detect outliers. System practicality is studied under the current 5G/6G capabilities.**

***Index Terms*—Autonomous driving, Trajectory prediction, Connected and autonomous vehicles (CAVs), Edge architecture, Clustering**

I. Introduction

Connected and autonomous vehicles (CAVs), are becoming increasingly prevalent, with the ambition of achieving full autonomy and given the complexity of such a mission, recent researches adopt an approach that involves breaking down the task into different main stages: scene perception, trajectory prediction and motion planning, as mentioned in [1], while scene perception and planning primarily focus on gathering scene information and selecting feasible routes respectively, the objective of trajectory prediction is to forecast the future positions of all the entities in the scene in order for the connected autonomous vehicle to plan safely in a dynamic environment. With the development of communication technologies, vehicles are able to share information with one another and the surrounding infrastructures, therefore including the connectivity perspective results in more reliable trajectory prediction as in [2] [3]. Accordingly we introduce our edge-assisted trajectory prediction, as illustrated in Fig.1, that uses clustering for the purpose of improving the motion prediction capabilities of CAVs. The following is a summary of the main contributions of this work:

- We present a new edge based architecture for trajectory prediction and confirm its viability for practical applications.
- We achieve state-of-the-art (SOTA) results on the real-world nuScenes vehicles dataset [4].
- We demonstrate the reliability of our method against anomalies.

To present our results in a clear and organized manner, the rest of the paper is structured as follows:

- In section II we review the related work.
- In section III we describe the functioning of each step of our method.
- In section IV we unveil the edge assistance architecture and the estimated traffic flow.
- Finally, in Section V we conclude and outline prospects for future research.

II. Related work

In this section, we review the recent work employed for trajectory prediction followed by a highlighting of relevant methods used for clustering trajectories.

This research was partially supported by Labex DigiCosme (project ANR11LABEX0045DIGICOSME) operated by ANR as part of the program ≪ Investissement d'Avenir ≫ Idex ParisSaclay (ANR11IDEX000302)

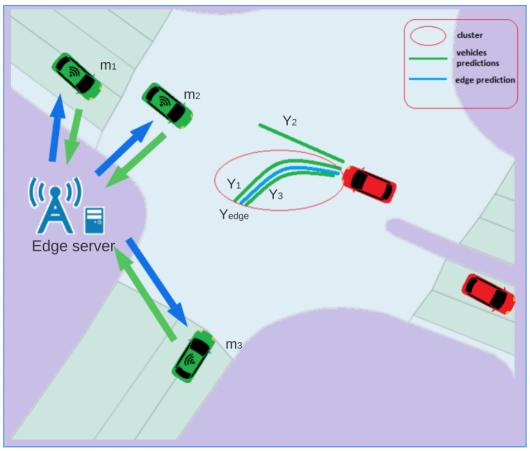

Fig. 1: illustration of our edge based trajectory prediction architecture, the CAVs predicts future trajectory of the target agent(red vehicle), predictions are sent to the edge server to preform the clustering, the final output is sent back to all CAVs in the scene

Trajectory prediction

Classical methods employed physical models, such as in [5], where they use a constant velocity model for motion prediction in a collision warning system. Another approach, as adopted in [6], uses a model that assumes a constant turn rate and acceleration. While those methods offer the advantage of computing trajectories quickly, their rigid assumptions have a detrimental impact on their predictive accuracy. In [7] Kalman filters are employed for a collision-free trajectories prediction and in [2] the authors proposed to use V2V communications [8] along side with Kalman filters to predict trajectories of connected cars allowing the ego car to forecast future motions of vehicles.

Recently with the advancement of machine and deep learning methods, a plethora of works have proposed models to predict trajectories. In [9] recurrent networks, more precisely, Long Short Term Memory (LSTM) based approach is used for predicting the motion of surrounding vehicles to the ego autonomous car where each agent's LSTM learns its state and predicts his future motion and in [10] they are applied to predict future maneuvers in an encoder-decoder network. The authors of [11] implemented visual attention and in [12] a graph attention network is used in order for the model to only attend for the most important information in the input. In [13] a spatially-aware graph neural network is introduced where interactions between agents are modeled, in the work of [14] they introduce a Conditional Variational Autoencoder (CVAE) generative model to generate future trajectories for the agents in the scene, in [15] vehicles are modeled as a clique to jointly predict multiple trajectories, a comprehensive study can be found in [16].

Trajectories clustering

During clustering, a collection of items is divided up into smaller groups using a certain similarity metric [17]. In the context of trajectory data, many similarity metrics are commonly employed. Notable examples include: Euclidean distance, Hausdorff distance and LCSS distance as presented in [18], these metrics are used to cluster trajectories according to the degree of their similarity.

After defining a similarity metric, a clustering method is applied as shown in Fig. 2. In [19], [20] the authors proposed a trajectory clustering method based on the DBSCAN and HDBSCAN algorithms respectively in an unsupervised manner. Whereas, in [21], the preferred approach of the authors is to use supervised clustering where vehicles trajectories are clustered with machine learning methods leveraging the learned vehicle vectors from the low-dimensional representations.

III. METHOD DESCRIPTION

We aim to predict the future positions Y_{edge} of each agent within a scene based on their past lateral and longitudinal positions X along with their acceleration, heading and map information.

First, the historic positions of the target agent are fed into CAVs trajectory prediction base models $(m_1, m_2, \ldots, m_n)$ to predict $(Y_1, Y_2, \ldots, Y_n)$. Next, these outcomes are transmitted to an edge server that calculates the similarity matrix of all predicted trajectories then clusters them using DBSCAN algorithm, the cluster with the most trajectories is averaged to obtain Y_{edge} which is then broadcast to all CAVs in the scene. Fig.3 illustrates the functioning of our method and algorithm 1 displays the Pseudo-code of our trajectory prediction. Further elaboration on the calculation of the similarity matrix and clustering is provided below.

Algorithm 1 Pseudo-code of our trajectory prediction algorithm

Input: Past history of agent X
Output: Predicted future trajectory $\mathbf{Y}_{edge}$
  $models \leftarrow$ list of models
  for $m \in models$ **do**
    $Y \leftarrow$ m.predict(X)
    Sendtoserver(Y)
    /* Each CAVs model sends it's predictions to server */
  end for
  $\mathbf{Y}_{all} \leftarrow (Y_1, Y_2, \ldots, Y_n)$, a set of n predicted trajectories from models $(m_1, m_2, \ldots, m_n)$
  $Sim\_matrix \leftarrow$ CalculateSimilarityMatrix$(\mathbf{Y}_{all})$
  $clusters \leftarrow$ DBSCAN$(Sim_matrix, \epsilon,$ min_samples$)$
  $Y_{edge} \leftarrow$ average(argmax$(clusters).values)$
  /* The cluster with most trajectories is averaged */
  Send_to_all(Y_{edge})
  /* Final prediction is broadcast to all CAVs */

Similarity matrix calculation

The similarity matrix is used as input to our clustering algorithm. To calculate it, we use the euclidean distance as our metric where each trajectory is represented by a set of points with varying length depending on the prediction horizon H. Concretely, each element of the similarity matrix consists of the pairwise distance between the trajectories.

Clustering algorithm

Trajectory clustering can be expressed mathematically as follows:
Let $C = \{C_1, C_2, \ldots, C_p\}$ be the set of P clusters, given the set of trajectories $Y_{all} = \{Y_1, Y_2, \ldots, Y_n\}$ the objective is to find P partitions using similarity metrics, where P is the number of clusters such that each data point Y_j belongs to exactly one cluster C_i.

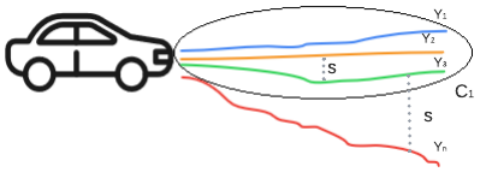

Fig. 2: Representation of trajectories clustering where S represents the similarity metric and C_1 is the produced cluster

Several clustering algorithms could be considered. Following recommendations in [22], our choice went for the DBSCAN [23]. We wanted a robust and powerful algorithm that does not need a prior knowledge of the number of clusters whereas in supervised learning labeled data is needed or the number of neighbors is predefined as in the KNN algorithm.

DBSCAN is a clustering algorithm that creates clusters of trajectories given the Similarity matrix as input, the minimum number of samples of each cluster and a parameter ϵ which defines the distance where points within it are added to the cluster. One key advantage of this algorithm is that the number of neighbors is not defined as hyper-parameter and the algorithm will create the clusters only depending on the distance between trajectories. We conduct more tests to select the best hyper-parameters in the experiments section.

IV. Edge Assistance Architecture

We explain here the designed edge architecture. Since data prediction concerns all the vehicles in a given confined area, we assume that each important road exchange location is equipped with an edge architecture such as the one we propose hereafter.

A. Data flow

Data flow is orchestrated by vehicles transmitting their ego view of all the surrounding other targets. Each target is located by its position. The data flow is continuous because predictions

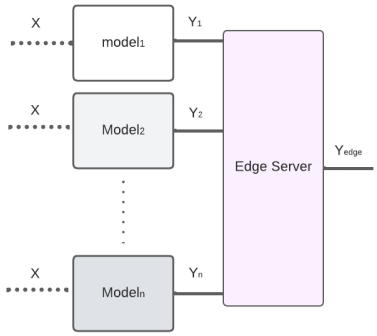

Fig. 3: Our method's architecture where historical data of the predicted vehicle are fed into each CAVs model, the outputs are sent to the edge server to obtain the final predicted trajectory

have to be made as explained before on a regular per second basis until the ego vehicle leaves the edge area. This leads to a huge amount of data that can only be tolerated in real time by 5G/6G capabilities. This justifies also the necessity of a localized edge service rather than a global centralized system that could not make the real-time predictions for large vehicular traffic zones.

- The edge broadcasts its position and pertinent information based on any protocol (it can be through D2D protocols or through vehicular DSRC systems).
- A point to point simple protocol is sufficient for the upstream data (from vehicle to edge). The vehicles segment their scene and send the data vectors of observed past 5 seconds. Vehicles with advanced capabilities can send their own predictions of the observed targets in the ego scene. Other vehicles with less sophisticated features can just send the observed trajectories of neighboring vehicles or even a frame by frame observation of the front and side views, the server will leverage this information with the help of its built-in models to generate new predictions if required.
- The scene is cut into several sectors (areas) according to the topographical and road regulations information.
- The edge gathers at least three predictions per scene sector. If it receives less than three, it uses its own camera of the scene and the data received from the non-sophisticated vehicles to make predictions and complete the rule of three different information.
- The clustering and prediction algorithm explained above are then used to calculate the next three seconds of car traffic flow.
- As real-time and safety are the key issues in such algorithms, results are broadcast to all the concerned agents in the scene.
- The results are then utilized by the CAVs to complete their autonomous maneuvers. Table I presents an estimated time

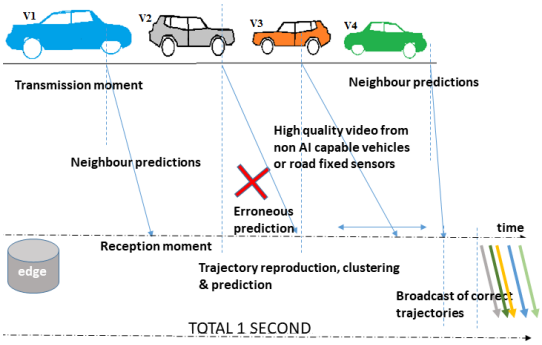

Fig. 4: Transmission, treatment and broadcast edge timeline

and throughput results depending on the number of agents in a sector.

Cloud Timeline: Our architecture provides a safe clustering prediction. The Fig. 4 shows as an example four vehicles that are connected to the cloud server when they approach the traffic zone. Some vehicles have high AI capabilities and can hence make their own predictions for the neighboring agents. Some other vehicles just send their ego view of the scene but do not make any prediction of future agent positions. Lastly, a subset of vehicles might have erroneous calculations for various reasons, yet they still transmit their predictions to the cloud server. As shown in the Fig. 4, the transmission delays from vehicle to the cloud server vary. This is due to the difference in amount of data sent (arrays of predictions per agent or real-time video), and also due to the difference in network interface speed that can vary from a vehicle to another (we give estimates of these limits in Table I). The cloud server receives all the information from different vehicles and eventually from its own fixed sensors on the scene. It executes all the procedures described in the previous section. Correct results are structured, grouped and sent back in a broadcast to the scene. Driving applications can also be updated through a web access. Each vehicle then receives data and filters its own pertinent information.

The timeline of all this procedure is very short as we need to repeat it continuously with present cars and new cars arriving to the scene. As an example, if we predict 3 future seconds based on passed 4 seconds. We have to skip the last second of events and all the process needs to be accomplished in one second. Table I explains the lower and higher bounds that each vehicle can send per second and every second to make the system feasible in the real life. Any delayed information is simply neglected by the server. The server sends back data in few nanoseconds. But as the receivers have different receiving speeds, it has to adapt to the slower reception rate. This is why we have different values in the downstream delay time. All this process has to be repeated in a pipeline manner.

TABLE I: Estimated traffic/delay at edge premises according to vehicle capabilities

Nb vehicles	upstream low	upstream high	downstream
5 vehicles	$10MB/vehicle$	$50MBytes/v.$	10 nanosec
10 vehicles	$4MB/vehicle$	$20MBytes/v.$	50 nanosec
100 vehicles	$300KB/vehicle$	$1.8MBytes/v.$	150 nanosec

V. EXPERIMENTS

In this experiments section we chose to employ 3 different base models, all are variations of Trajectron++ due to its proven superior predictions and the availability of its source code, the first version is the basic model that doesn't include dynamic integration, the second version includes the latter to produce more realistic and Dynamically-Feasible trajectories and finally the variant that also includes map information instead of only using tracked trajectories which the official best Trajectron++ model that is used to compare against other models. It is important to point out that all these base models have inferior results than our method. For the clustering algorithm the minimum number of samples to create a cluster is set to 1 and more tests are done to find the best ϵ.

A. Dataset

To accurately evaluate our method we used the nuScenes dataset which consists of around 10^5 vehicles and more than 1000 driving scenes, each one containing 20 seconds. The scenarios are annotated using human experts. To compare with the nuScenes Vehicles Motion Prediction SOTA models the same dataset composition for evaluation is used as in [14] and [15].

B. Metrics

To evaluate the performance of our trajectory prediction method, we employ the commonly used ADE and FDE metrics.

- Average displacement error (ADE): represents the average L2 distance in meters where the trajectories are evaluated over the entire prediction horizon is calculated as in [16] with the following:

$$ADE = \frac{1}{N_A \times H} \sum_{i=1}^{N_A} \sum_{t=1}^{H} \left| \hat{Y}_t[i] - Y_t[i] \right| \quad (1)$$

  Where N_A is the number of agents in the scene, H is the prediction horizon and $\hat{Y}$, Y are respectively the predicted and the ground truth trajectories over H.

- Final displacement error (FDE): represents the final L2 distance in meters where the trajectories are evaluated over the final predicted position T, it is calculated as follows:

$$FDE = \frac{1}{N_A} \sum_{i=1}^{N_A} \left| \hat{Y}_T[i] - Y_T[i] \right| \quad (2)$$

Our experiments were conducted using a server equipped with an NVIDIA Quadro RTX 4000 GPU, Intel(R) Xeon(R) Gold CPU, 755GiB of Ram and Ubuntu 22.04.2 LTS system.

C. Results

In the following we compare our method to the SOTA trajectory predictions models.

We present the results of our experiments in Table II and III which display, the results of FDE and ADE, respectively. Our method gives best performance in comparison with SOTA models.

TABLE II: FDE results (the lower the better) show that our method achieves SOTA performance.

Method	@1s	@2s	@3s	@4s
S-LSTM [9]	0.47	–	1.61	–
CSP [10]	0.46	–	1.50	–
CAR-Net [11]	0.38	–	1.35	–
spAGNN [13]	0.35	–	1.23	–
Trajectron++ [14]	**0.07**	0.45	1.14	2.20
SCEPT(bestof 3) [15]	0.40	0.80	1.36	2.14
Ours	0.08	**0.43**	**1.08**	**2.03**

TABLE III: ADE results (the lower the better) show that our method achieves SOTA performance.

Method	@1s	@2s	@3s	@4s
Trajectron++	**0.07**	0.19	0.45	0.82
Ours	0.08	0.19	**0.43**	**0.77**

Longer prediction horizons: To confirm the hypothesis that our method gives significantly better improvement for longer prediction horizons, we render the results with comparison to the best model in Fig. 5 and Fig. 6 then calculate the improvement percentage, results confirms that when prediction time is longer our method improves the predicted trajectories further in comparison to single models.

Epsilon value selection: One key hyper-parameter of our clustering algorithm is epsilon. It determines the size of the neighborhood for the clustering. To ascertain the most suitable value for epsilon, we carried out several experiments and documented the outcomes in Table IV where it shows that the best performance is obtained when epsilon=10.

TABLE IV: FDE results with different epsilon values.

Epsilon	@1s	@2s	@3s	@4s
eps=0.1	**0.07**	0.44	1.12	2.15
eps=1	0.08	0.44	1.09	2.10
eps=10	0.08	0.43	1.08	**2.03**
eps=100	0.08	0.43	1.08	2.04

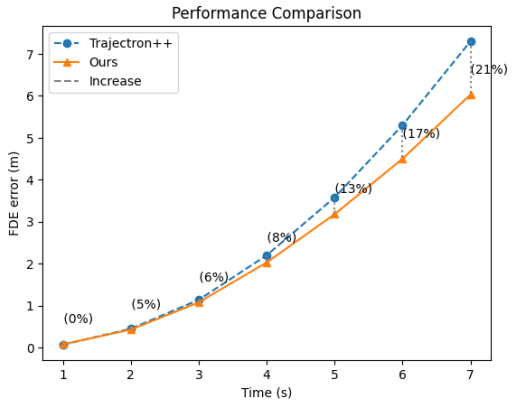

Fig. 5: FDE for longer prediction horizon

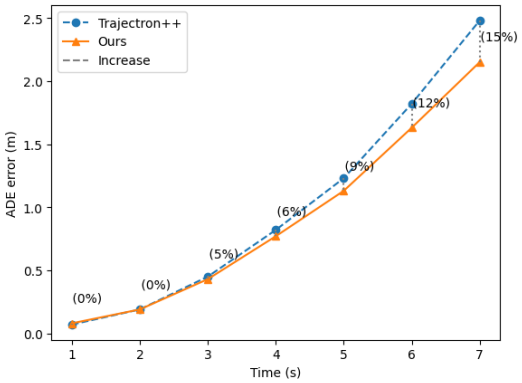

Fig. 6: ADE results

Failure robustness: To efficiently study the robustness of our system to anomalies, we describe in the following a scenario where a system failure occurs to one of the agents causing it's predictions to be utterly erroneous, as shown in Fig. 7, this is done by using an inadequately trained prediction model that produces bad predictions. Moreover to the ADE/FDE errors and to get a better understanding of the final prediction accuracy, we additionally add the miss rate metric which expresses if the trajectory hit the final destination point in a rayon of d meters, a prediction is considered as a miss if it's further than the threshold d, in our experiments we set $d = 2$.
In Table V we can see that our method gives good results even with the presence of failures due to its ability to cluster trajectories and detect outliers.

VI. Conclusion and Future Work

A crucial aspect for achieving full autonomy is integrating connectivity and data exchange into trajectory prediction which remains relatively unexplored. To fill this gap, we designed, implemented and Carried out evaluations of a novel edge based trajectory prediction architecture for connected autonomous

TABLE V: FDE/MR/ADE results shows that our method achieves good performance even with a failure model.

Time	@1s			@2s			@3s			@4s		
Metrics	FDE	MR	ADE	FDE	MR	ADE	FDE	MR	ADE	FDE	MR	ADE
Failure	3.53	14%	2.62	7.04	16%	7.04	10.62	17%	6.16	14.31	18%	7.94
Our	**0.09**	**0%**	**0.08**	**0.44**	**1%**	**0.44**	**1.12**	**10%**	**0.44**	**2.15**	**13%**	**0.80**

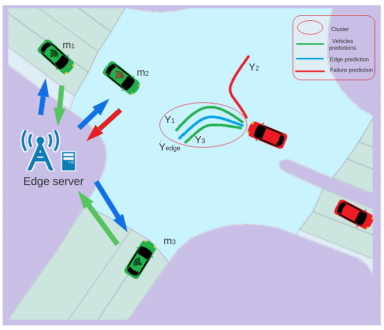

Fig. 7: Illustration of the failure agent robustness scenario. Good final prediction output is obtained even with an abnormal agent

vehicles. Our method is capable of achieving state-of-the-art results while robustly dealing with anomalies.

To demonstrate the viability of our architecture, we estimated the data traffic under the current 5G/6G capabilities.

Investigating more advanced clustering methods Suggests a potential optimization of clusters formation ultimately leading to enhance the results. Furthermore, due to the fundamental Impact on Cluster Formation exploring a wider range of similarity metrics holds a promising potential prospect for future research .

REFERENCES

[1] W. Zeng, W. Luo, S. Suo, A. Sadat, B. Yang, S. Casas, and R. Urtasun, "End-to-end interpretable neural motion planner," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8660–8669.

[2] R. Zhang, L. Cao, S. Bao, and J. Tan, "A method for connected vehicle trajectory prediction and collision warning algorithm based on v2v communication," *International Journal of Crashworthiness*, vol. 22, no. 1, pp. 15–25, 2017. [Online]. Available: https://doi.org/10.1080/13588265.2016.1215584

[3] M. Baek, D. Jeong, D. Choi, and S. Lee, "Vehicle trajectory prediction and collision warning via fusion of multisensors and wireless vehicular communications," *Sensors*, vol. 20, no. 1, 2020. [Online]. Available: https://www.mdpi.com/1424-8220/20/1/288

[4] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *CVPR*, 2020.

[5] R. Miller and Q. Huang, "An adaptive peer-to-peer collision warning system," in *Vehicular Technology Conference. IEEE 55th Vehicular Technology Conference. VTC Spring 2002 (Cat. No.02CH37367)*, vol. 1, 2002, pp. 317–321 vol.1.

[6] A. Polychronopoulos, M. Tsogas, A. J. Amditis, and L. Andreone, "Sensor fusion for predicting vehicles' path for collision avoidance systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 8, no. 3, pp. 549–562, 2007.

[7] V. Lefkopoulos, M. Menner, A. Domahidi, and M. N. Zeilinger, "Interaction-aware motion prediction for autonomous driving: A multiple model kalman filtering scheme," *IEEE Robotics and Automation Letters*, vol. 6, no. 1, pp. 80–87, 2021.

[8] A. Demba and D. P. F. Möller, "Vehicle-to-vehicle communication technology," in *2018 IEEE International Conference on Electro/Information Technology (EIT)*, 2018, pp. 0459–0464.

[9] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 961–971.

[10] N. Deo and M. M. Trivedi, "Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1179–1184.

[11] A. Sadeghian, F. Legros, M. Voisin, R. Vesel, A. Alahi, and S. Savarese, "Car-net: Clairvoyant attentive recurrent network," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 151–167.

[12] X. Mo, Z. Huang, Y. Xing, and C. Lv, "Multi-agent trajectory prediction with heterogeneous edge-enhanced graph attention network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 9554–9567, 2022.

[13] S. Casas, C. Gulino, R. Liao, and R. Urtasun, "Spatially-aware graph neural networks for relational behavior forecasting from sensor data," *arXiv preprint arXiv:1910.08233*, 2019.

[14] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVIII 16*. Springer, 2020, pp. 683–700.

[15] Y. Chen, B. Ivanovic, and M. Pavone, "Scept: Scene-consistent, policy-based trajectory predictions for planning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 103–17 112.

[16] Y. Huang, J. Du, Z. Yang, Z. Zhou, L. Zhang, and H. Chen, "A survey on trajectory-prediction methods for autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 652–674, 2022.

[17] M. Y. Ansari, Mainuddin, A. Ahmad, and G. Bhushan, "Spatiotemporal trajectory clustering: A clustering algorithm for spatiotemporal data," *Expert Systems with Applications*, vol. 178, p. 115048, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417421004899

[18] J. Bian, D. Tian, Y. Tang, and D. Tao, "A survey on trajectory clustering analysis," 2018.

[19] H. Xu, Y. Zhou, W. Lin, and H. Zha, "Unsupervised trajectory clustering via adaptive multi-kernel-based shrinkage," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 4328–4336.

[20] D. Zhang, K. Lee, and I. Lee, "Hierarchical trajectory clustering for spatio-temporal periodic pattern mining," *Expert Systems with Applications*, vol. 92, pp. 1–11, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0957417417306450

[21] W. Wang, F. Xia, H. Nie, Z. Chen, Z. Gong, X. Kong, and W. Wei, "Vehicle trajectory clustering based on dynamic representation learning of internet of vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3567–3576, 2021.

[22] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu, "Dbscan revisited, revisited: Why and how you should (still) use dbscan," *ACM Trans. Database Syst.*, vol. 42, no. 3, jul 2017. [Online]. Available: https://doi.org/10.1145/3068335

[23] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proceed-*

ings of the Second International Conference on Knowledge Discovery and Data Mining, ser. KDD'96. AAAI Press, 1996, p. 226–231.