Motion Prediction Of Traffic Agents With Hybrid Recurrent-Convolutional Neural Networks

Vasileios Lagoutaris, Konstantinos Moustakas, Senior Member, IEEE
Electrical and Computer Engineering Department, University of Patras, Greece

***Abstract*—Self-driving vehicles are expected to dramatically reduce road accidents and improve the quality of life of millions of people. However, they still have a long way to go before matching the performance of the best human drivers. In this paper we tackle the problem of motion prediction of traffic agents. Our proposed solution combines recurrent and convolutional neural networks to model the spatial and temporal interactions between agents. It is based on the encoder-decoder architecture that can model the past motion of traffic agents, which it improves by leveraging features extracted from a Bird's Eye View image of the driving scene for its predictions. We train and evaluate our model in the largest publicly available self-driving dataset for motion prediction. Our model achieves better performance than approaches based on standalone recurrent neural networks or standalone convolutional neural networks respectively.**

***Index Terms*—motion prediction, multimodal trajectory prediction, recurrent neural networks, convolutional neural networks**

I. Introduction and Related Work

DRIVING a motorized vehicle on the road is a very challenging task, considering it requires rapidly adapting to a constantly changing environment with road actor motion prediction capabilities. In this section we provide a review of existing work on the field of motion prediction, considering works based on recurrent neural networks and convolutional neural networks separately.

A. Methods based on Recurrent Neural Networks

Recurrent neural networks were one of the first types of neural networks to be used for motion prediction in traffic scenarios. This is the case because vanilla RNNs with a sufficient number of hidden units can approximate any finite sequence to sequence mapping [1]. A LSTM network is used in [2] to predict the future longitudinal and lateral trajectories of vehicles on a highway, as a regression task. A LSTM network is also used in [3] to predict the future positions of multiple vehicles at the same time in an occupancy grid. This LSTM receives the positions, velocities and yaw rates for all the considered vehicles. Some other methods use additional RNNs to either output multimodal trajectories or model the interactions between vehicles. Dai *et al.* [4] propose a spatio-temporal LSTM-based trajectory prediction model that uses two different groups of LSTMs. Different types of LSTMs are used in [5] to output multimodal trajectories and their probabilities.

Although RNNs have proven to be very successful in sequence to sequence tasks due to their ability to model temporal data, they struggle to capture other important elements of motion prediction, such as the interactions between moving agents and the geometry of the driving environment.

979-8-3503-3959-8/23/$31.00 ©2023 IEEE

B. Methods based on Convolutional Neural Networks

Researchers have tried to represent their data as images to be processed by a CNN. A common way of accomplishing this goal in the motion prediction for traffic scenarios task is to use a BEV image as the input to the motion prediction model. A simplified BEV image that describes the locations of vehicles and lanes is used in [6] to make predictions about intention to change lane. A CNN with the convolution-deconvolution architecture [7] is used to output an occupancy map in [8]. Casas *et al.* [9] use 2 different CNNs to process a rasterized map and voxelized BEV data in parallel. The extracted features of the 2 CNNs are concatenated and then 3 different heads are added on top to detect vehicles, estimate their intention, and predict their trajectory by the same backbone network.

CNNs can receive images as input in the form of BEV images of the traffic scene or output images in the form of an occupancy map. In [10] researchers use convolutional networks to solve the problem of motion prediction for traffic agents using BEV images of the traffic scene. In contrast to RNNs, they are capable of capturing spatial relationships in the data, which is a prerequisite to model the interactions between moving agents and take into account driving context. Their drawback is their inability to model temporal dependencies.

C. Motivation and Contributions

Few approaches to motion prediction for traffic scenarios have managed to successfully combine recurrent and convolutional networks. Mukherjee *et al.* [11] feed an Occupancy Grid Map to a modified LSTM, to predict the future positions of all cars in the scene. Multi-head attention is used in [12] after considering a joint representation of the static scene and surrounding agents. Our approach expands this area by efficiently combining spatial and temporal features of the driving scene, which are fed to an LSTM-based decoder that can accurately predict multiple future trajectories for the target agent. The aforementioned advancements are provided as open-source to the scientific community.

II. Hybrid motion prediction

In this section we present our proposed hybrid method for predicting multiple trajectories for traffic agents along with their probabilities. First, we define the problem of motion

prediction for moving traffic agents on the road and the used notation, then we discuss our proposed approach including the network architecture and the optimization function used.

A. Problem definition

Let us denote the discrete times at which the perception system outputs state estimates as $T = \{t_1, t_2, \ldots, t_T\}$. Also, we denote the state estimate of the i-th agent at time t_j as S_{ij}, where $i = 1, 2, \ldots, N_j$, with N_j being the number of unique agents picked up by the perception system at time t_j. Naturally, the number of unique agents perceived at each time-step differs as new agents come within proximity of the vehicle's sensors and others move further away.

In addition to the state estimates, through the dataset we have access to a semantic map R. This semantic map includes information about the number of lanes, the direction of each lane and which lane connects to which. In an additional layer of detail, precise lane geometry is included. With the use of the state estimates as well as the semantic map, we can construct a BEV image of the target agent's surrounding environment, which includes static and dynamic components.

We want to predict the future behavior of certain moving agents that interest us, which we call target agents (TAs). Their behavior can be described by their predicted future state estimates $\hat{S}_{TAs} = \{\hat{S}_{i(n+1)}, \hat{S}_{i(n+2)}, \ldots, \hat{S}_{i(n+K)}\}_{i=1}^{N_n}$, where K is the prediction window, meaning the amount of time steps for which we are interested in predicting the agents' behavior (we denote the current time step as t_n). Without loss of generality and for simplicity's sake, in this work we limit ourselves to the prediction of the target agent's future x- and y-positions instead of their full state estimates. The full state estimates of the target agents can be derived using the known agents' states along with the predicted positions. So, instead of $\hat{S}_{TAs}$, we want to predict $\hat{x}_{TAs} = \{\hat{x}_{i(n+1)}, \hat{x}_{i(n+2)}, \ldots, \hat{x}_{i(n+K)}\}_{i=1}^{N_n}$ and $\hat{y}_{TAs} = \{\hat{y}_{i(n+1)}, \hat{y}_{i(n+2)}, \ldots, \hat{y}_{i(n+K)}\}_{i=1}^{N_n}$ respectively.

Essentially, taking the probabilistic nature of the problem into account, we want to compute the conditional distribution

$$P\left(\hat{x}_{TAs}, \hat{y}_{TAs} | S_n, R\right). \tag{1}$$

Here R is the semantic map, while S_n corresponds to the full state estimates of all N_n agents up to and including the current time step t_n, with H being the number of historic time steps considered:

$$S_n = \left\{S_{i(n-H)}, S_{i(n-H+1)}, \ldots, S_{i(n)}\right\}_{i=1}^{N_n}. \tag{2}$$

In simpler terms we want to compute the conditional distribution of the agents' future x- and y-positions, given the agents' current and past state estimates along with the semantic map. Since eq. (1) is a mutual distribution over series of positions of several independent agents, it can be intractable. To reduce the computational requirement, we treat each target agent separately and compute the following conditional distribution:

$$P\left(\hat{x}_A, \hat{y}_A | S_n, R\right), \tag{3}$$

where $\hat{x}_A = \{\hat{x}_{A(n+1)}, \hat{x}_{A(n+2)}, \ldots, \hat{x}_{A(n+K)}\}$ and $\hat{y}_A = \{\hat{y}_{A(n+1)}, \hat{y}_{A(n+2)}, \ldots, \hat{y}_{A(n+K)}\}$ and A is the selected target agent.

B. System description

We assume access to the historic state estimates of the target agent A for the time steps $\{t_{n-H}, t_{n-H+1}, \ldots, t_{n-1}\}$ as well as its state estimate for the current time step t_n. All these state estimates together are denoted by S_A, which can be expressed as follows:

$$S_A = \left\{S_{A(n-H)}, S_{A(n-H+1)}, \ldots, S_{A(n)}\right\}. \tag{4}$$

Here, H represents the number of historic state estimates that are considered for the prediction. The state estimates S_A are coming from a perception system after it processes the output of the ego-vehicle's sensors. In the purposes of our model, the state estimate $S_{A(j)}$, which refers to the state estimate of the target agent A at the time step t_j, contains the following information:

$$S_{A(j)} = \left\{x_{A(j)}, y_{A(j)}, \dot{x}_{A(j)}, \dot{y}_{A(j)}, \varphi_{A(j)}, a_{A(j)}\right\}, \tag{5}$$

where:

- $\{x_{A(j)}, y_{A(j)}\}$: are the target agent's x- and y-position
- $\{\dot{x}_{A(j)}, \dot{y}_{A(j)}\}$: are the target agent's velocities in the longitudinal and lateral directions respectively
- $\varphi_{A(j)}$: is the target agent's yaw in radians
- $a_{A(j)}$: is a binary value indicating the availability of the aforementioned information for the time step t_j

The future trajectories of the target agent are represented by their x- and y-positions over time. We want to make predictions for the future trajectory of the target agent spanning K time steps into the future so essentially, we want to predict $x_A = \{x_{A(n+1)}, x_{A(n+2)}, \ldots, x_{A(n+K)}\}$ and $y_A = \{y_{A(n+1)}, y_{A(n+2)}, \ldots, y_{A(n+K)}\}$. To address the multimodality associated with motion prediction, the model produces multiple separate future trajectory predictions. Each mode is represented by its own index and the output of the model is M possible future trajectories $\hat{x}_A = \{\hat{x}^m_{A(n+1)}, \hat{x}^m_{A(n+2)}, \ldots, \hat{x}^m_{A(n+K)}\}_{m=1}^{M}$ and $\hat{y}_A = \{\hat{y}^m_{A(n+1)}, \hat{y}^m_{A(n+2)}, \ldots, \hat{y}^m_{A(n+K)}\}_{m=1}^{M}$ along with their probabilities p_m, such that $\sum_{m=1}^{M} p_m = 1$. Here m indicates the mode index and K as explained is the prediction window. With this approach we formulate the problem of motion prediction as a regression problem to the future trajectories of the TA.

To approximate the conditional distiribution (3) and make predictions about the future trajectory of a particular TA, the sequence of the last $H + 1$ state estimates is fed into the encoder of the model. The final cell state and hidden state produced by the encoder form the encoder vector, which encapsulates the state estimate input sequence. At the same time the state estimates S_A are used to rasterize a BEV image of the driving scene. We experimented with a simple 3-channel RGB BEV image that contains only semantic information,

as well as a 17-channel image that additionally contains the positions of the target agent and the surrounding agents for 6 time steps of history, in seperate channels. We call the first approach Semantic-Hybrid and the second Dynamic-Hybrid. A convolutional neural network is then used to extract features from this image, which are concatenated with the encoder vector to combine spatial and temporal features.

Our hybrid model received the state estimates $S_{A(j)}$ as given by eq. (5) for H historic time steps. In cases where the historic information of the target agent consisted of less than H time steps, the availability indicator $\alpha_{A(j)}$ was given the value of zero for the time steps whose historic information was absent. This allowed the model to generalize and to be able to make accurate predictions even with less historic information, which is the case in the validation and test sets. Furthermore, data standardization is adopted in the state estimates S_A that are fed to the encoder, since the input features exist on wildly different scales.

C. Network architecture

The detailed network architecture, which is explained in detail in the following paragraphs, is illustrated in Fig. 1.

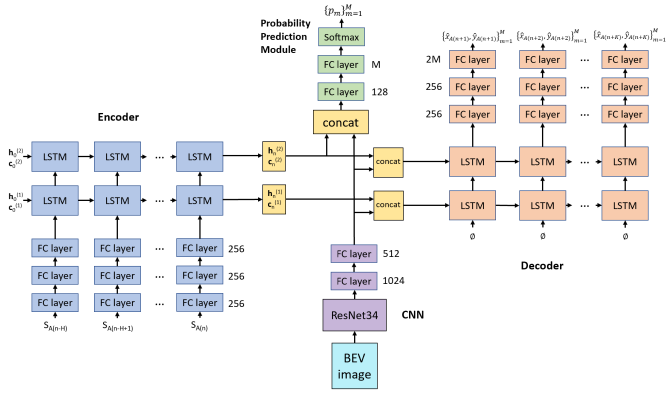

Fig. 1: The detailed network architecture of the hybrid approach.

1) Encoder-Decoder: The historic and current state estimates of the target agent S_A first pass through some fully connected layers, to match the dimensions of the LSTM modules and to allow the network to capture the complex structure of the state estimate data. We use 3 fully connected layers for this purpose, with all hidden layers as well as the output dimension being equal to 256. A ReLU activation function follows each layer's linear output. After H recursive updates in the two LSTMs, their last hidden and cell states $\boldsymbol{h}_n^{(1)}$,$\boldsymbol{c}_n^{(1)}$ and $\boldsymbol{h}_n^{(2)}$,$\boldsymbol{c}_n^{(2)}$ form the encoder vector and are used by the decoder as well as the probability prediction module.

The decoder has a similar architecture to the encoder, consisting of 2 stacked LSTM layers followed by 3 fully connected layers. the last one has a dimension of $2M$. That is because the output of the last fully connected layer is the final prediction of the model for the M modes of x- and y-positions of the target agent at the corresponding time step.

2) Convolutional neural network: The convolutional neural network has been added to the encoder-decoder architecture as a feature extractor. The BEV rasterized image of the driving scene is directly fed to the CNN backbone, which after some experimenting is chosen to be a ResNet34 [13]. The first 4 layers of the network are followed by an Average Pooling layer, while the final fully connected layer of the ResNet34 is omitted. The backbone CNN is followed by 2 fully connected layers with output dimensions of 1024 and 512 respectively. The ResNet34 that is used as a backbone feature extractor is pretrained for classification on ImageNet.

Both the Semantic-Hybrid and the Dynamic-Hybrid models use the architecture illustrated in Fig. 1. The only difference is that modifications need to be made to the ResNet34 in the case of the dynamic-hybrid model, since it is designed to handle only 3-channel images by default. Specifically, the input dimension of its first convolutional layer is increased to match the number of image channels.

3) Probability Prediction Module: The probability prediction module consists of 2 fully connected neural network layers. A ReLU activation function is included as the non-linearity in the first fully connected layer, while the second layer does not contain a non-linear activation function. The output of the second fully connected layer is however passed through the Softmax function, to ensure that the predicted probabilities sum up to one.

D. Optimization function

Assuming that the ground truth positions are modelled by a mixture of multi-dimensional independent Normal distributions over time, yielding the likelihood:

$$\sum_m p_m \, \mathcal{N}\left(x_{n+1,\dots,n+K} \middle| \hat{x}^{1,\dots,M}_{n+1,\dots,n+K}, \Sigma = 1\right) \mathcal{N}\left(y_{n+1,\dots,n+K} \middle| \hat{y}^{1,\dots,M}_{n+1,\dots,n+K}, \Sigma = 1\right)$$

$$= \sum_m p_m \prod_t \mathcal{N}\left(x_t | \hat{x}_t^m, \sigma = 1\right) \mathcal{N}\left(y_t | \hat{y}_t^m, \sigma = 1\right) . \quad (6)$$

This gives us the following NLL loss for the particular target agent A:

$$NLL_A = -\log \textstyle\sum_m e^{\log p_m - \frac{1}{2}\sum_t \left(\hat{x}^m_{A(t)} - x_{A(t)}\right)^2 + \left(\hat{y}^m_{A(t)} - y_{A(t)}\right)^2} . \quad (7)$$

Our network tries to minimize the corresponding loss function over all training examples, resulting in the minimization of:

$$\sum_i NLL_i, \quad (8)$$

where i iterates over all training examples.

Since the resulting loss function (7), is a Log-Sum-Exp function (LSE) [14], we use the log-sum-exp trick for numerical stability. This trick is used to improve accuracy and

avoid underflow and overflow problems when dealing with very small or very large numbers. The formula is:

$$LSE(x_1, \ldots, x_n) = x^* + \log\left(e^{x_1 - x^*} + \ldots + e^{x_n - x^*}\right), \quad (9)$$

where $x^* = max\{x_1, \ldots, x_n\}$.

III. Experimental Results

For our experiments we used the largest publicly available dataset for motion prediction, which is provided by Lyft Level 5 [15]. We used 5 seconds of historic state estimates resulting in 50 timestamps.

We compare the Semantic-Hybrid and Dynamic-Hybrid models to the following baselines: (i) Encoder-Decoder model, by removing the CNN from the hybrid model and only feeding it with the state estimates of the target agent and (ii) Multiple-Trajectory Prediction (MTP) [10].

A. Comparative evaluation

The performance of the models in some of the most commonly used metrics for motion prediction are presented in Table I. These include the negative log likelihood (NLL) of the ground truth given each model's predicted trajectories, the displacement between the ground truth and its closest predicted trajectory at 1s and 5s (same as final displacement error). They also include the average displacement error oracle, meaning as calculated by only considering the closest mode to the ground truth and the weighted average displacement error (WADE), which weights all modes according to their predicted probability.

TABLE I: Performance of the models on the most common motion prediction metrics for the validation dataset.

		Displacement(m)			
Method	**NLL**	**@1s**	**@5s**	**ADE**	**WADE**
Enc-Dec	51.15	0.1774	1.2681	0.6097	1.3114
MTP	25.17	0.1564	**0.6541**	0.3879	0.8600
Sem-Hyb	22.31	0.1469	0.7846	0.4231	0.9826
Dyn-Hyb	**16.49**	**0.1331**	0.6623	**0.3718**	**0.8008**

As we can observe, the standalone encoder-decoder model that does not use a CNN has significantly worse performance on all motion prediction metrics. This highlights the performance uplift that can be gained by feeding the scene context to a motion prediction model. The Semantic-Hybrid model achieves 11% lower NLL compared to the MTP model, while the Dynamic-Hybrid model outperforms it by a massive 34%. On displacement based metrics, the Semantic-Hybrid model can not keep up with the MTP model, while the Dynamic-Hybrid model surpasses it in all metrics apart from the final displacement error.

B. Comparisons per agent type

The Lyft Level 5 dataset contains multiple agent types. More spesifically, the target agents in the dataset can be vehicles, cyclists or pedestrians. Table II presents the negative log likelihood of all trained models, broken down by the type of agent whose trajectory they are predicting. This way comparisons can be made about the relative accuracy of the predictions between different agent types. It can be observed that all models achieve roughly identical performance in vehicle motion prediction as they do when including all agent types. This is expected as vehicles form the vast majority of agents in the dataset. Furthermore, every model produces higher NLL scores when predicting the future motion of cyclists. The relative difference between the NLL scores of vehicles and cyclists seems to increase for better performing models, with the dynamic-hybrid model producing almost double NLL scores when predicting the future motion of cyclists.

TABLE II: The negative log likelihood loss of all models broken down by agent type.

	NLL			
Method	**Total**	**Vehicles**	**Cyclists**	**Pedestrians**
Enc-Dec	51.15	51.25	70.88	19.13
MTP	25.17	25.19	38.16	7.27
Sem-hyb	22.31	22.36	**31.15**	7.62
Dyn-hyb	**16.49**	**16.36**	31.27	**6.02**

As far as pedestrians are concerned, their future trajectory seems to be easier to predict as every model achieves lower NLL scores for them. This is partly because pedestrians travel shorter distances when compared to the rest of the agents within the 5s prediction window. The negative log likelihood metric as defined by eq. (7), assumes unit variance, meaning it takes lower values for shorter trajectory predictions.

IV. Conclusions

Motion prediction of moving agents on the road is an important prerequisite of self-driving. In this work we presented two variations of a hybrid approach that combines recurrent and convolutional neural networks. Our hybrid models were able to perform very well on a variety of traffic scenarios and produced low-error 5s predictions. The importance of giving a detailed representation of the traffic scene to a motion prediction model is highlighted in comparative experiments.

Given the multimodal nature of driving, a route planning system could leverage all modes of predicted trajectory for agents surrounding the self driving vehicle to generate the final route it should follow.

Although our subsampling process ensured that the training examples given to the model were not similar to each other, future research could focus on the improvement of the subsampling procedure, which could lead to faster convergence.

V. Acknowledgements

This work has received funding from the European Union's Horizon 2020 research and innovation program under Grant Agreement No. 101094364 - ITHACA: Artificial Intelligence to Enhance Civic Participation.

REFERENCES

[1] B. Hammer, "On the approximation capability of recurrent neural networks," *Neurocomputing*, vol. 31, no. 1, pp. 107–123, 2000. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231299001745

[2] F. Altché and A. de La Fortelle, "An lstm network for highway trajectory prediction," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 353–359.

[3] S. H. Park, B. Kim, C. M. Kang, C. C. Chung, and J. W. Choi, "Sequence-to-sequence prediction of vehicle trajectory via lstm encoder-decoder architecture," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, 2018, pp. 1672–1678.

[4] S. Dai, L. Li, and Z. Li, "Modeling vehicle interactions via modified lstm models for trajectory prediction," *IEEE Access*, vol. 7, pp. 38 287–38 296, 2019.

[5] 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.

[6] D. Lee, Y. P. Kwon, S. McMains, and J. K. Hedrick, "Convolution neural network-based lane change intention prediction of surrounding vehicles for acc," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 2017, pp. 1–6.

[7] H. Noh, S. Hong, and B. Han, "Learning deconvolution network for semantic segmentation," in *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1520–1528.

[8] S. Hoermann, M. Bach, and K. Dietmayer, "Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 2056–2063.

[9] S. Casas, W. Luo, and R. Urtasun, "Intentnet: Learning to predict intention from raw sensor data," *CoRR*, vol. abs/2101.07907, 2021. [Online]. Available: https://arxiv.org/abs/2101.07907

[10] H. Cui, V. Radosavljevic, F.-C. Chou, T.-H. Lin, T. Nguyen, T.-K. Huang, J. Schneider, and N. Djuric, "Multimodal trajectory predictions for autonomous driving using deep convolutional networks," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 2090–2096.

[11] S. Mukherjee, S. Wang, and A. Wallace, "Interacting vehicle trajectory prediction with convolutional recurrent neural networks," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 4336–4342.

[12] K. Messaoud, N. Deo, M. M. Trivedi, and F. Nashashibi, "Trajectory prediction for autonomous driving based on multi-head attention with joint agent-map representation," in *2021 IEEE Intelligent Vehicles Symposium (IV)*, 2021, pp. 165–170.

[13] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[14] P. Blanchard, D. J. Higham, and N. J. Higham, "Accurately computing the log-sum-exp and softmax functions," *IMA Journal of Numerical Analysis*, vol. 41, no. 4, pp. 2311–2330, 08 2020. [Online]. Available: https://doi.org/10.1093/imanum/draa038

[15] J. Houston, G. Zuidhof, L. Bergamini, Y. Ye, A. Jain, S. Omari, V. Iglovikov, and P. Ondruska, "One thousand and one hours: Self-driving motion prediction dataset," *CoRR*, vol. abs/2006.14480, 2020. [Online]. Available: https://arxiv.org/abs/2006.14480