Multi-agent Approach to Predict the Trajectory of Road Infrastructure Agents Using a Convolutional Neural Network

Andrey Azarchenkov [1] and Maksim Lyubimov [1]

[1] *Bryansk State Technical University, 7, 50-let Oktyabrya bul., Bryansk, 241035, Russia*

Abstract

The problem of creating a fully autonomous vehicle is one of the most urgent in the field of artificial intelligence. Many companies claim to sell such cars in certain working conditions. The task of interacting with other road users is to detect them, determine their physical properties, and predict their future states. The result of this prediction is the trajectory of road users' movement for a given period of time in the near future. Based on such trajectories, the planning system determines the behavior of an autonomous-driving vehicle. This paper demonstrates a multi-agent method for determining the trajectories of road users, by means of a road map of the surrounding area, working with the use of convolutional neural networks. In addition, the input of the neural network gets an agent state vector containing additional information about the object. A number of experiments are conducted for the selected neural architecture in order to attract its modifications to the prediction result. The results are estimated using metrics showing the spatial deviation of the predicted trajectory. The method is trained using the nuscenes test dataset obtained from lgsvl-simulator.

Keywords

Trajectory prediction, Autonomous-driving car, Convolutional neural network, Nuscenes dataset, lgsvl simulator.

1. Introduction

Driving a vehicle is a complex and many-sided task, since it requires real-time analysis of the surrounding and making a decision to perform a particular maneuver. In addition to its complexity, human errors cause serious consequences for both the driver and others. A potential way to minimize driving risks in the future is to autonomous-driving vehicles.

The main idea of creating an unmanned vehicle is to develop a software package that can analyze the surrounding of the car and make driving decisions, depending on the result of the analysis. Nowadays the developers of such systems distinguish 6 levels of autonomy of unmanned vehicles [15], where the zero level corresponds to vehicles without automation, and the fifth – to fully autonomous cars. With the increase in the level of autonomy, the question of predicting the position of the surrounding vehicles becomes more acute. The actions of other road users are often variable and difficult to predict. To solve this problem, software systems for driving a car in one form or another have a subsystem for predicting the states of objects [16].

2. Related works

The problem of predicting trajectories can be solved using various types of information. The most popular group of methods is based on working with sequences of positions of objects in space. Most often, such methods are used to predict the trajectories of pedestrians. ETH [1] and UCY [6] data sets can be distinguished. ETH dataset contains 2 scenes taken from a bird's-eye view and contains a total

GraphiCon 2021: 31st International Conference on Computer Graphics and Vision, September 27-30, 2021, Nizhny Novgorod, Russia
EMAIL: azarchenkovaa@gmail.com (A. Azarchenkov); max32@inbox.ru (M. Lyubimov);
ORCID: 0000-0003-4570-4442 (A. Azarchenkov); 0000-0003-0702-3662 (M. Lyubimov);

of 750 different motion paths. In this case, the distance between the positions in space is determined with a step of 0.4 seconds. UCY dataset contains 3 scenes, also shot from a bird's eye view. In total, it contains 900 trajectories with a position change step of 0.4 seconds. Usually, these two data sets are used together, and often one of the five scenes is selected for testing, and methods using recurrent layers are usually applied [2, 5].

Prediction methods differ from each other in the way they provide information about the future positions of objects. Most of methods, including the work discussed above, generate one or more object trajectories. However, other methods of prediction are also possible. For example, in [18] the result of the neural model is the area where the object can be, and the color indicates the time after which it will be there (Figure 1). To obtain such a grid map, the surrounding space is divided into cells of a fixed size and the probability of finding an object in each of these cells is indicated. Joint prediction of objects trajectories is also possible [19]. In this case, in addition this approach takes into account not only the positions of other objects, but also their intentions.

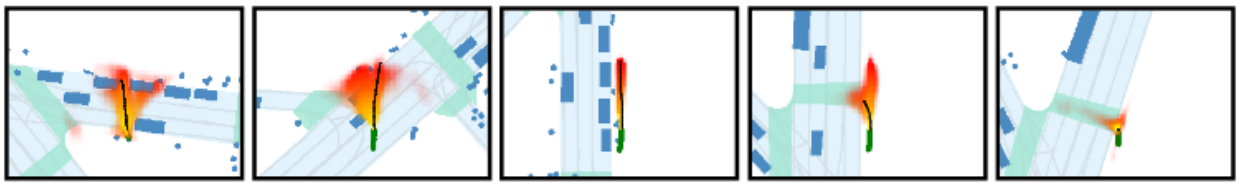

Figure 1: An example of grid maps generated

This paper describes a method for solving the problem in the field of autonomous vehicles. The peculiarity of the task is that the prediction must be made for road users (agents) of different types. At the same time, it is important that the positions of all surrounding objects are taken into account when making the prediction. This approach is called multi-agent. Another feature of this field is that today the common practice for unmanned cars is the use of pre-prepared maps, which are referred to hd maps [14]. Most often, they are used for the movement of the unmanned car itself, but since these maps repeat public roads, their use allows to predict the behavior of agents more accurately. In [13], the prediction of cars behavior was based on the use of the map of the surrounding area.

For the prediction system, such maps are very often presented in a bitmap format. If it is necessary to make prediction, an image is created for the agent, which indicates its position on the map, as well as the position of other agents. In this case, objects of different types are highlighted in different colors. In addition, for each object its previous states are plotted at a fixed point in time. Thus, using the previous states, you can evaluate the speed of the object. Such an image is used, for example, in CoverNet neural network [4] and is called input representation. The advantage of this approach is the ability to use convolutional layers, which perfectly solve the problem of highlighting key features in the image. However, there is also an approach in which the input representation is given as a vector [3]. The authors used graph neural network, which input contained map elements and the trajectories of agents as vectors. The vectors formed polylines, each group was initially processed by the neural network separately, and then a fully connected graph was used, based on the fact that the features are connected to each other.

3. Method

The main idea of the prediction method is to generate an image containing the object under study, other objects, as well as a map of the area. Such an image is referred to as an input representation. Due to the fact that all agents are added to the input representation, the approach can be considered multi-agent. Figure 2 shows an example of such an image. This input representation is fed to the input of the convolutional neural network.

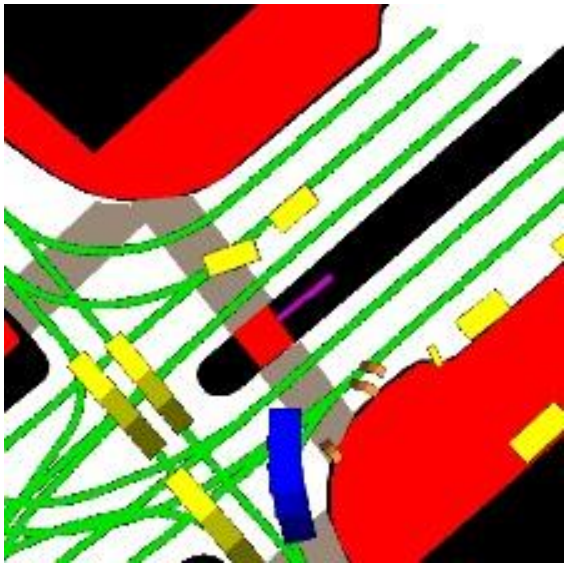

Figure 2: An example of input representation

RepVGG architecture is used as the main convolutional network (backbone) [7]. A distinctive feature of this architecture is its simplicity and speed. This convolutional network has different states at the time of training and at the time of using the model (Figure 3). At the time of training, the network has a residual structure, which is used primarily in ResNet models [8]. The idea of such a structure is to increase the accuracy of deep neural networks, since without a residual mechanism, the error of the network increases with increasing its depth. The network contains convolutional blocks, relu activation function, batch normalization, and layer concatenation. After training, the neural model can be converted into a simple network that does not have branching. At the same time, the network variant for applying the model contains only 3x3 convolutional blocks and ReLU activation function. These layers are well optimized in Nvidia Cudnn [9] and Intel MLL [10] libraries, which allows to get the maximum performance of the model. This aspect is relevant, since the task of predicting movement trajectories is often performed for a large number of objects at the same time. In addition, after converting the model, the video memory consumed by it is reduced, which is also important for embedded systems.

After the main convolutional network, there is a block for processing the received features. Two methods were tested. In the first case, the feature matrix is reduced to a one-dimensional type, the result of the operation is fed to the input with linear layers, to extract the trajectory and confidence for each of them. The second method is to perform a weighted subsampling. Initially, the features are multiplied by the matrix, which values are selected as a result of training, then subsampling operation is performed.

The input of the neural model, in addition to the input representation, is also fed with a vector containing the speed, acceleration, and class of the object for which the prediction is made. Using such a vector, the neural network explicitly obtains the physical characteristics of the object. This vector is combined with the image features obtained after convolution (Figure 4).

4. Data preparation

Nuscenes data set was selected for training the neural model [11]. It consists of 1000 scenes, each lasting 20 seconds. Each scene is a set of data flows from sensors installed on the car, as well as an annotation for them. Cameras, lidars, radars, GPS and IMU sensors are used as these sensors. The entire dataset contains 1,400,000 images and 390,000 lidar images. The data set is characterized by a high-quality annotation, which is very important for predicting trajectories. The annotation contains a description of all the objects, and also maps of the surrounding area. To work with the data set, the developers provided a library that allows to access objects and recorded data.

In addition to this data set, our own data were collected on the basis of the open software package for driving a car – Apollo [17], as well as LGSVL simulator [12]. The main idea of creating our own data set is to use the results of the surrounding object detection system to generate an annotation to input representations. LGSVL simulator allows to receive messages in a format similar to Apollo scene system. Apollo software package is based on the cyber framework, which gives the opportunity to form and transmit messages between separate independent modules. To get our own data set, a program was made that reads such messages and creates input representations based on them. In this case, the future states of the detected object were used as an annotation. Since the correct determination of the object's position was important for this task, only those objects detected by means of lidar were used.

As a result of comparing the data in nuscenes training sample and in our own sample, it turned out that the average speed of all moving cars in our own sample is 6 m/s, while in nuscenes data set it is 2 m/s. This comparison indicates the need to collect own training sample when using the neural model in practice.

For further training, it was decided to combine the collected samples, since the developed data set contains new patterns of vehicle behavior. As part of the work, we used the vertical reflection of the input representation. In addition to reflecting the input representation itself, the true trajectory of the object also changes. The tests performed (Table 3) showed that these augments have a positive effect on the result.

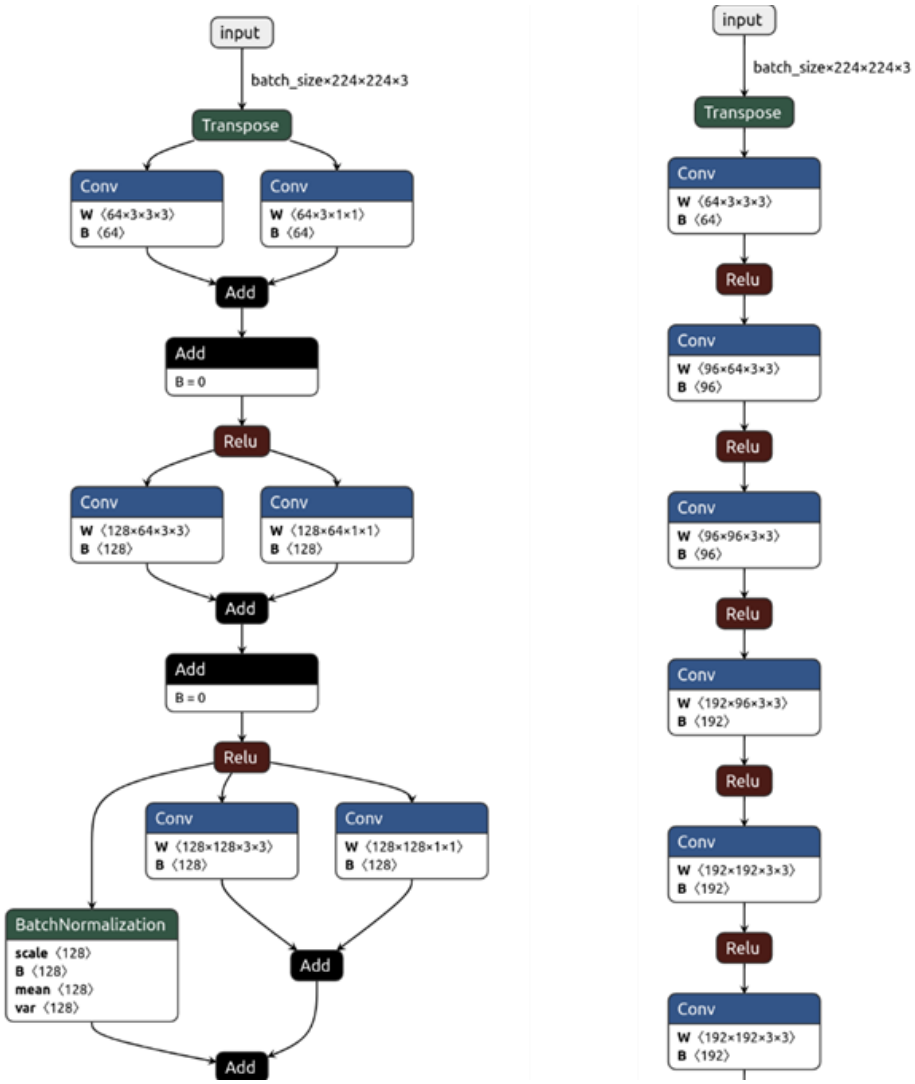

Figure 3: Structure of a neural model while training (left) and inference (right)

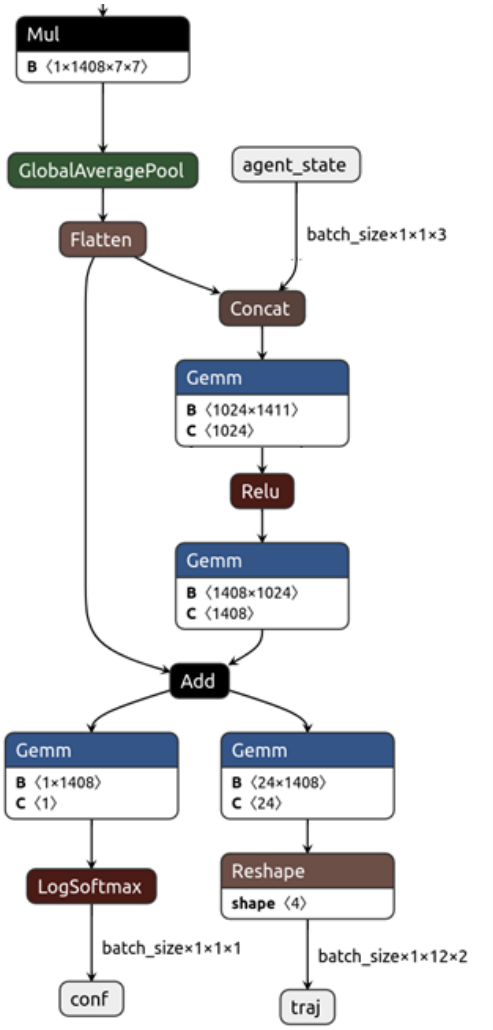

Figure 4: Configuration of input layers of the neural model

5. Experiments

To solve the prediction problem, a number of experiments were conducted with different configurations of the neural model and training parameters. To verify the results, the neural model was validated on a pre-prepared training sample, to which all moving cars, pedestrians, and cyclists from Nuscenes test sample were added, resulting in 53,154 input representations.

The following metrics were used to evaluate the quality of the neural model:

- average displacement error (ADE) is the average distance between two separate points of the trajectory expressed in meters (1);

$$ADE = \sum_{i=1}^{12} \sqrt{(X_i - x_i)^2 + (Y_i - y_i)^2} \quad (1)$$

where X, Y – ground truth coordinates, x, y – predicted coordinates.

- final displacement error (FDE) is the distance between two final points expressed in meters (2);

$$FDE = \sqrt{(X_{12} - x_{12})^2 + (Y_{12} - y_{12})^2} \quad (2)$$

where X, Y – ground truth coordinates, x, y – predicted coordinates.

All experiments were conducted sequentially and the best candidate was selected at each stage. The network was trained to predict possible trajectories, and the metrics were calculated using all predictions simultaneously. Figure 5 shows an example of such a prediction.

According to the results of testing the available architectures (Table 1), A2 model was chosen for further consideration, since according to the results obtained, the operating time of the models increases from A0 to B2, respectively. Since time is an important factor for this task, we gave preference to faster architecture instead of accuracy. Table 2 shows a comparison of metrics for a trained model with and without augmentation. Since vertical augmentation had a positive effect on the result, we decided to conduct further experiments with it. The additional information vector (Table 3) also had a positive effect on the result. The final test was conducted to identify the optimal number of trajectories (Table 4). According to the results, an increase in the number of trajectories has a positive effect on the result. However, when the algorithm is actually used in a prediction system, excessive trajectories can cause a lot of uncertainty for the planning system. Thus, a decision was made to stop at 9 trajectories, since the growth of metrics does not compensate for the increasing uncertainty in practice. Figure 5 shows the result of the network operation. To make it clear, the picture contains the trajectory that has the highest probability.

Table 1
Comparison of test set metrics for different neural network architectures

Architecture	ADE	FDE	Inference time(ms)
RepVgg-A0	1.153	2.537	1.0021
RepVgg-A1	1.131	2.485	1.2499
RepVgg-A2	1.112	2.429	2.2184
RepVggB0	1.102	2.392	2.9513
RepVggB1	1.097	2.365	3.9749
RepVggB2	1.057	2.299	5.7855

Table 2
Comparison of test set metrics for different augmentation ways

Augmentation	ADE	FDE
RepVgg-A2 without augmentation	1.112	2.429
RepVgg-A2 with augmentation	1.088	2.365

Table 3
Comparison of test set metrics for approaches with using additional data and without them

Using additional data	ADE	FDE
RepVgg-A2 without additional data	1.088	2.365
RepVgg-A2 with additional data	1.07	2.324

Table 4
Comparison of test set metrics for approaches with different number of predictions

A number of predicted trajectories	ADE	FDE
RepVgg-A2 with 1 predicted trajectory	1.07	2.324
RepVgg-A2 with 3 predicted trajectories	0.664	1.355
RepVgg-A2 with 5 predicted trajectories	0.53	1.013
RepVgg-A2 with 7 predicted trajectories	0.5	0.926
RepVgg-A2 with 9 predicted trajectories	0.467	0.829
RepVgg-A2 with 11 predicted trajectories	0.438	0.744
RepVgg-A2 with 13 predicted trajectories	0.417	0.688

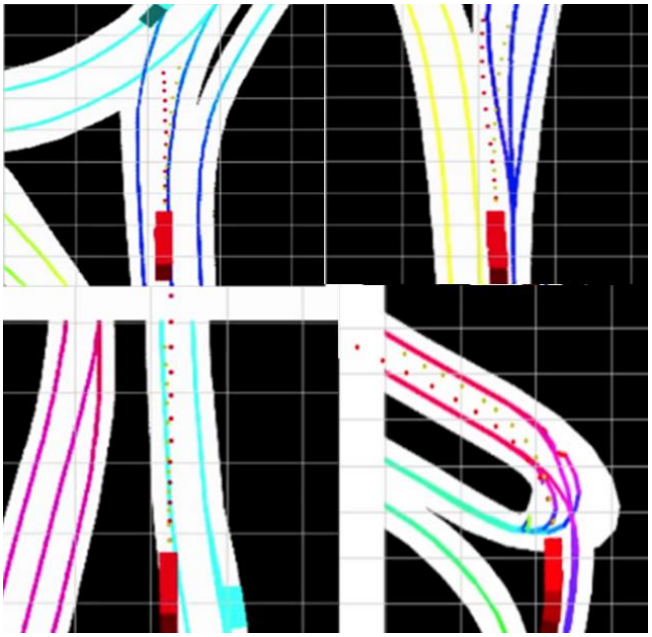

Figure 5: An example of movement trajectory prediction

6. Conclusion

The paper considers the problem of predicting the trajectories of road network agents. To solve this problem, we used a multi-agent approach with an input representation to make a prediction. A convolutional neural network was designed that is suitable for this task in terms of speed and accuracy, nuscenes training sample was analyzed, and our own data were collected to expand it. The resulting solution was tested taking into account various possible modifications and network configurations. On the basis of the tests it was found that the applied approach to predicting trajectories does not have such a significant dependence on the depth of the neural model, compared to classical problems that use convolutional neural networks, for example, in the task of detecting objects. In addition, an important observation is that the resulting neural network is able to determine the type of object implicitly and make a prediction in accordance with the behavior patterns characteristic of this type. Also, the neural model successfully takes into account other agents when making a prediction. The resulting metrics allow us to judge that the described approach can be successfully applied in real systems for driving an unmanned car.

7. References

[1] S. Pellegrini, A. Ess, K. Schindler, L. van Gool, You'll Never Walk Alone: Modeling Social Behavior for Multi-Target Tracking. In: 2009 IEEE 12th International Conference on Computer Vision. Sept. 2009, pp. 261–268. doi:10.1109/ICCV.2009.5459260.

[2] N. Ma, Yuexin TrafficPredict: Trajectory Prediction for Heterogeneous Traffic. arXiv:1811.02146v5, 2019. – 8 p. doi:10.1609/aaai.v33i01.33016120

[3] J. Gao, C. Sun, H. Zhao, Y. Shen, D. Anguelov, C. Li, C. Schmid, VectorNet: Encoding HD Maps and Agent Dynamics from Vectorized Representation. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2020, pp. 11525-11533. doi: 10.1109/CVPR42600.2020.01154

[4] E. Grigore, F. Boulton, O. Beijbom, E. Wolff, CoverNet: Multimodal Behavior Prediction using Trajectory Sets 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020, pp. 14062-14071. doi: 10.1109/CVPR42600.2020.01408
[5] Z. Wenjing, Trajectory Prediction with Recurrent Neural Networks for Predictive Resource Allocation. 14th IEEE International Conference on Signal Processing (ICSP), 2018 – 1-8 p.
[6] A. Lerner, Y. Chrysanthou, D. Lischinski, Crowds by Example. Computer Graphics Forum (2007), pp. 655–664.
[7] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, J. Sun, RepVGG: Making VGG-style ConvNets Great Again arXiv preprint arXiv:2101.03697
[8] K. He, X. Zhang, S. Ren, J. Sun, Residual learning for image recognition. InProceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016. doi: 10.1109/CVPR.2016.90
[9] S. Chetlur, C. Woolley, P. Vandermersch, J. Cohen, J. Tran, B. Catanzaro, B. Shelhamer, cudnn: Efficient primitives for deep learning, arXiv preprint arXiv:1410.0759, 2014.
[10] Intel oneAPI Math Kernel Library, 2020. URL: https://software.intel.com/content/www/us/en/develop/tools/math-kernel-library.html.
[11] Nuscenes dataset, 2020. URL: https://www.nuscenes.org.
[12] SVL Simulator, 2021. URL: https://www.svlsimulator.com/.
[13] A. Azarchenkov, M. Lyubimov, Algorithm for predicting the trajectory of road users to automate control of an autonomous vehicle CEUR Workshop Proceedings, 2020, 2744
[14] S. Casas, A. Sadat, R. Urtasun, MP3: A Unified Model to Map, Perceive, Predict and Plan. arXiv:2101.06806 2021
[15] Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," SAE J3016, 2016. URL: https://www.sae.org/standards/content/j3016_201806/.
[16] The Autoware Foundation, 2020. URL: https://www.autoware.org/.
[17] Apollo autonomous driving, 2020. URL: https://apollo.auto/.
[18] A. Jain, S. Casas, R. Liao, Y. Xiong, S. Feng, S. Segal, R. Urtasun Discrete Residual Flow for Probabilistic Pedestrian Behavior Prediction, arXiv:1910.08041v1.
[19] A. Cui, A. Sadat, S. Casas, R. Liao, R. Urtasun, LookOut: Diverse Multi-Future Prediction and Planning for Self-Driving, arXiv:2101.06547v1, 2020