End-to-End learning for Context-Aware Multi-agents Trajectory Prediction

Hoonhee Cho
Department of Robotics Program
gnsgnsgml@kaist.ac.kr

Saeyoon Oh
School of Computing
saeyoon17@kaist.ac.kr

Kunho Kim
Department of Electrical Engineering
kaist984@kaist.ac.kr

Hyojin Sim
School of Computing
gywls655@kaist.ac.kr

You can see the detailed code of our project at **our github repository** or `https://github.com/Chohoonhee/CS470-TEAM4-TERM-PROJECT`

Abstract

Self-driving cars have received lots of attention for several years. People nowadays started to actually believe in autonomous driving systems. Yet, we must not forget that there are still problems. Meanwhile, Trajectory Prediction has been found to play a crucial role in monitoring self-driving cars. Although there have been some approaches toward improving prediction, we believe the performance is yet not so well as to use in real-time prediction. Therefore through the research, we tend to build a novel model for ourselves to tackle this subject.

1. Introduction

As the importance of self-driving cars rises, so as the importance of tools to monitor the autonomous driving system. The task of trajectory prediction aims to predict the spatial coordinate of various road-agents such as cars, pedestrians, animals, etc. If the prediction can be done in real-time, this technique will allow preventing the upcoming accident. Through the paper, we introduce a novel approach to predict the best path possible.

2. Related Work

2.1. Multimodal Trajectory Prediction

While a vanilla trajectory prediction aims to predict a path for an agent, Multimodal Trajectory Prediction model predicts multiple paths at the same time[1]. This was the paper that we started with and was one of the baseline for our study. The model uses CNN with FCN to produce multimodal prediction. Yet, there are two big problems of the model. First is that since the model architecture is too simple, it does take the context into account. Therefore, it tends to produce predictions which sometimes are not plausible. Also the loss function optimizes only the best path among multiple trajectories which makes other trajectories more non-plausible. Through our model we tried to overcome these problems by changing loss function and using more complex model.

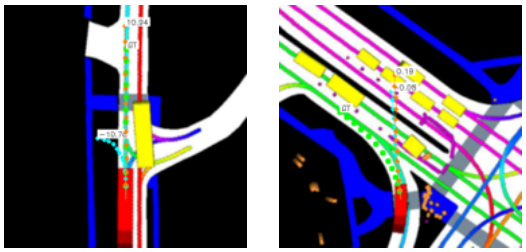

Figure 1: Results of Multimodal Trajectory Prediction

2.2. CoverNet

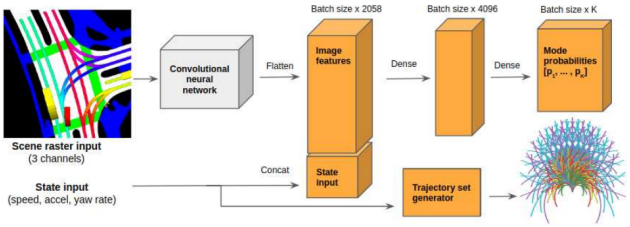

Figure 2: Architecture of CoverNet

CoverNet is one of the famous trajectories papers published since MTP using the nuScences dataset[3]. This model uses the generator to extract various modes, instead of using

MLP to extract probabilities for each mode. The advantage of using this method is that more modes can be extracted. However, the problem with this method is that the proper path does not come out even if the loss is applied to various modes. Due to the lack of understanding of image context, the road may be violated or the path may be incorrect.

2.3. R2P2 RNN

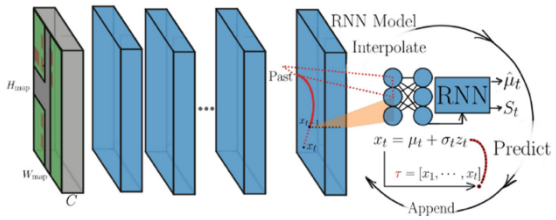

Figure 3: Architecture of R2P2 RNN

R2P2 is a paper that used the KITTI dataset to solve the projectory task, although it did not use the nuScenes dataset[5]. This model uses a past path using Vanilla RNN and has a property in that it uses sequential data. However, the problem still exists that the output is not semantic because information about roads or lanes is not properly accepted.

3. Proposed Method

In this section, we present our thoughts that we had in mind while building our architecture. One big problem with the previous model was that it seemed to not capture the contexts. Therefore, one of the most important parts of our model was to carefully model our architecture to actually look at the context. There have been two approaches to handle this issue.

3.1. Using multiple agents

Previous model takes only one agent into account for the prediction. We believe the rationale behind this is that previous authors believed the interaction between agents can be done only with the image which turned out to be not true. Therefore, we now feed the past information about multiple agents directly into the model. This information is fed to the LSTM Encoder in which will be described in more detail in the next part. Through directly handing over this information and with LSTM and Attention between agents, we thought we would be able to capture the interaction between agents.

3.2. Separating the input images

The previous model inputs the whole Bird-Eye-View image as one raster image. Yet it is pretty intuitive that this kind of formation would not be good for the model since it then has to learn to also discretize the information (Agent, Path, Road information). Therefore we now divide one raster image into three different images. We then concatenate the vectors and feed them to the pre-trained CNN network.

3.3. Optimizing best path using modified loss

Another problem of the previous one dealt with in section 2.1 is that it optimizes just the single best-predicted path. The given loss function is as follows.

$$\mathcal{L}_{ij}^{TMP} = \mathcal{L}_{ij}^{class} + \alpha \sum_{m=1}^{M} I_{m=m*} L(\tau_{ij}, \hat{\tau}_{imj}) \quad (1)$$

$$\mathcal{L}_{ij}^{class} = -\sum_{m=1}^{M} I_{m=m*} \log p_{im} \quad (2)$$

We have confirmed through many examples that this approach generates only one persuasive path. Such that ($m = m_*$) Since the rationale of multiple trajectory prediction is to predict multiple feasible paths, this is not what we want for path prediction. Also since all other paths are bad, there is the possibility that with a small chance this could actually be an obstacle of prediction. Therefore we decided to work on a single best path. The modified loss function that we use is as follows.

$$\mathcal{L} = \sum_{n=1}^{|K|} [L(s_{n,t}^i, \hat{s}_{n,t}^i) + \alpha L(\dot{s}_{n,t}^i, \hat{\dot{s}}_{n,t}^i)] \quad (3)$$

Where K is the set of trajectory, s_t^i the coordinate of ith agent at the time-step t, $\dot{s}_t^i$ the difference between s_t^i and s_{t+1}^i divided by the time-step. (α is hyperparameter weight in which we choose.) First thing to note is that now we are optimizing for a single path. Also through weighted sum of the second term, we also made our model to also predict the velocity. (Note that the difference divided by time-step refers to velocity of an agent.)

4. Model Architecture

The backbone of our model CMTP is an LSTM model with attention. Figure 5 depicts the model, which includes an encoder, an attention-based decoder, and a CNN network. At a high-level, our model takes pre-processed segment images, past trajectory vector, and agent velocity as input and produces a prediction path of the agent, which are then converted to images. We describe these components below.

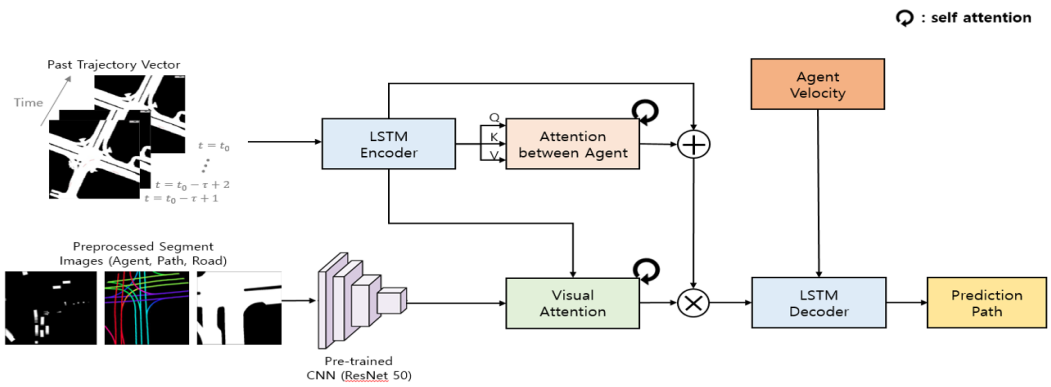

Figure 4: The Context-Aware Multi-agents Trajectory Prediction model structure

4.1. CNN MODEL

We used ResNet50 for CNN backbone. We did not use the image as a simple 3 channel so that the model could better understand the context of the image. Pre-trained CNN takes pre-processed segment images which are divided one raster image into three different images. Each image is an extract of the only agent, path, and the road from a base raster image. When experimenting with MobileNet, ResNet34, ResNet50, and ResNet150 did not differ much, but ResNet50 did the best.

4.2. LSTM ENCODER

The encoder takes the past trajectory vector T as input. We first initialize hidden state values to zero with (number of layers, number of total past trajectory vectors, hidden size = 32). We also create relative current predict sequence, which is zero values of same size with past trajectory vectors. It has only first sequence before embedding. After linear embedding that vectors, we resize it again and put padding in. After that using LSTM for predict objects trajectory [4].

$$\begin{aligned} &Output, (h_t, c_t) \\ &= LSTM(Embedded\ Object\ Trajectory, h_t - 1) \\ &= Encoder(T) \end{aligned} \tag{4}$$

4.3. VISUAL ATTENTION

Visual attention is based on the fact that when a person looks at a picture, he or she does not look at all the parts of the picture, but only looks at interesting parts. Visual attention was used to efficiently learn what should be noted in each image. Visual attention takes the first extracted the segment image feature Γ through convolutional neural network and the final encoding data as input. After operating of visual attention, we can get pixel-wise attention γ_t at each encoding step.

$$\gamma_t = \Gamma \odot Visual\ Attention(h_t, \Gamma, h_t) \tag{5}$$

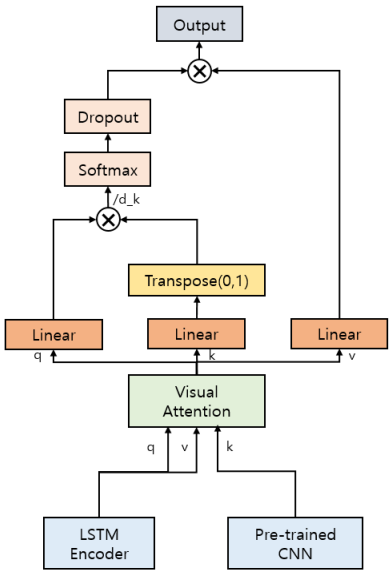

Figure 5: Visual attention

4.4. SELF ATTENTION

Self-attention is a method of calculating how many vectors are related to each other in image vectors and reflecting them in the model. In other words, self-attention can identify relationships between image vectors. We used Scaled Dot Product Attention for self-attention. The collection of the final encoding data is passed to the self attention module. After operating of self attention, we can get cross-agent representation $\tilde{h_t}$.

$$\tilde{h_t} = h_0 + Self\ Attention(h_t) \tag{6}$$

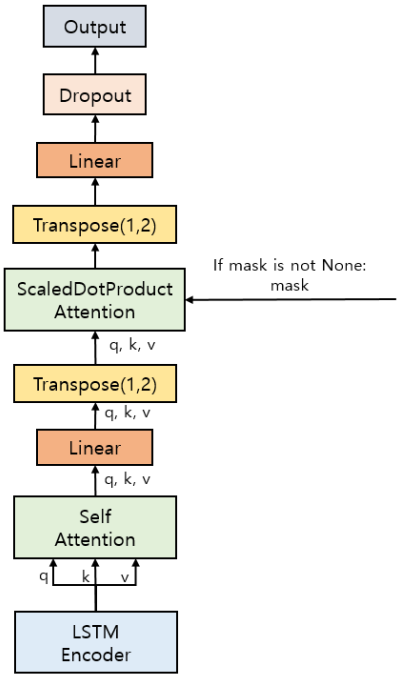

Figure 6: Self attention

4.5. LSTM DECODER

LSTM Decoder receives the results of the preceding encoding data, the start velocity of the agent. We first initialize hidden state c_t values to zero with (number of layers, batch size, hidden size = 32). Hidden state h_t has the start velocity of agent. We also create state tuple which has two hidden states. Circulating by the length of the sequence, the state tuple and the preceding encoded data are inserted into the decoder. After linear embedding that output, we stack up that results into predict list. Finally, we can get predict path of agent after permutation and cumulative sum of predict list [4].

$$\begin{aligned} &Predict\ Path, (h_t, c_t) \\ &= LSTM(Encoder(T)\ Output, (h_t - 1, c_t - 1)) \\ &= Decoder(Encoding\ Data, Start\ Velocity) \end{aligned} \tag{7}$$

5. Experimental Description

We implemented our models using pre-trained ResNet-50 as our backbone CNN. We read the ResNet Conv5 feature map and apply a global pooling layer. We just use the velocity stated vector as LSTM decoder inputs. And the learning rate was periodically reduced to one-tenth when the loss did not decrease compared to the validation loss continuously.

5.1. Dataset

We used nuScenes datasets that contain 850 different real-world driving scenarios, where each spanning 20 seconds of frames[1]. It also provides drivable-area maps and agent states. the number of datasets is limited to around 25K. So, we used the cropped images for each sequence to be 5 seconds. And using a sampling rate of 2 Hz that is a commonly used value in the trajectory dataset. We also employ Kalman smoothing to reduce noise and impute missing points in trajectories. We separate the dataset with 32,186 observations in the train set, 8,560 observations in the train-val set, and 9,041 observations in the validation set. This split publicly available in the nuScenes software development kit.

5.2. Metrics

There are two metrics in which we evaluate our model. These methods are taken from the methods used in the other papers and are one of the indicators for evaluating the performance of trajectories.

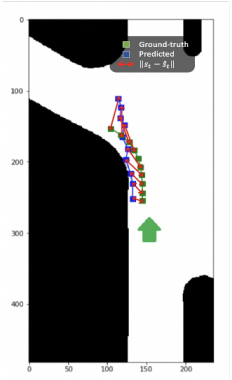

Figure 7: minADE

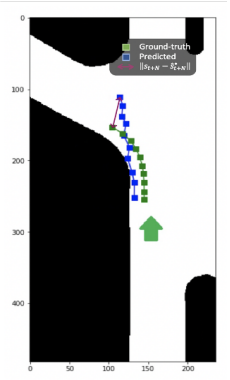

Figure 8: FDE

First one is $minADE$ (Minimum Average Displacement Error)

$$minADE_k = min_{\hat{s} \in \mathcal{P}} \frac{1}{N} \sum_{\tau=t}^{t+N} \|s_\tau - \hat{s}_\tau\| \tag{8}$$

The metric calculates the average displacement between the ground truth trajectory and the predicted trajectory, where s_τ is the location of an agent at predicted time-step τ and $\hat{s}_\tau$ is at ground truth time-stemp τ. Used as an indicator to evaluate the overall path, such as direction, was well generated.
Another metric is FDE (Final displacement Error) which analyzes the final displacement error.

$$FDE = \left\| s_{t+N} - \hat{s}^*_{t+N} \right\| \tag{9}$$

This value shows how well the endpoint of the path of the targeted agents was predicted. This value shows how well the endpoint of the path of the targeted agents was predicted.[3]

6. Results

6.1. Evaluation through the visualization

Our model can make predictions for different agents at the same time as below. We showed the result image for the three representative scenes that are rotation, intersection, and straight road. When compared to the past, the context of the image can be read well to show that it does not deviate from the road. Also, the length of the predicted path is accurate using sequence information.

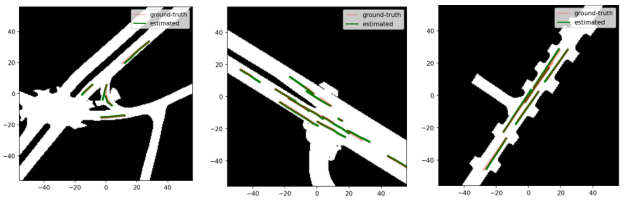

Figure 9: Our model prediction results

The causes of these results are thought to be the effects of the following four techniques.

Effectiveness of cross-agent interaction module: Path can be generated by considering the interaction of each other when generating path of the agent. Therefore, it is possible to create the correct path to generate by referring to other path without creating a conflicting or overlapping path.

Effectiveness of visual attention module: In order to properly generate path, image context is required for the image. Visual attention is a way to give the model information about the image. It actually kept agents from intruding into the non-drive area.

Effectiveness of LSTM encoder and decoder: LSTM encoder used information about past path to extract feature. [1] MTP did not use this sequence information, so semantic path could not be obtained. Also, decoder extracted path sequentially and made it a more suitable model for the path trajectory.

Effectiveness of adding the difference of loss function: Trajectory is a task that predicts displacement, but there is also a velocity information used. Therefore, it is possible to create a difference value using displacements to obtain the loss function. This loss also helps to predict a sudden increase or decrease in displacements.

6.2. Quantitative Evaluation

ADE, FDE metric was used for quantitative evaluation of the model. For ADE, the min value was used because some other models produced multiple pathways. In such a model, the value with the lowest ADE among the multi-mode pathways is expressed as minADE. As for our model, we used multiple paths for agents at a time and averaged out.

Metric	minADE ↓	FDE ↓
MTP	1.88	5.22
Covernet	2.77	6.65
R2P2	1.17	2.19
MATF	1.05	2.12
CMTP(OURS)	**1.07**	**2.2**

Table 1: Trajectory Model evaluation

We compared our model with various papers since MTP[2]. Compared to the high-performance R2P2[5] and MATF[6], it can be seen that the performance does not lag behind. This problem seems to be solved through decoder transformation. In addition, in the case of ADE, other models obtained min values among the various paths, but we gave a penalty to our model and evaluated it in that we generate one path. We can conclude based on the experimental results that the visual attention and agent attention we have created help improve performance in actual path trajectories.

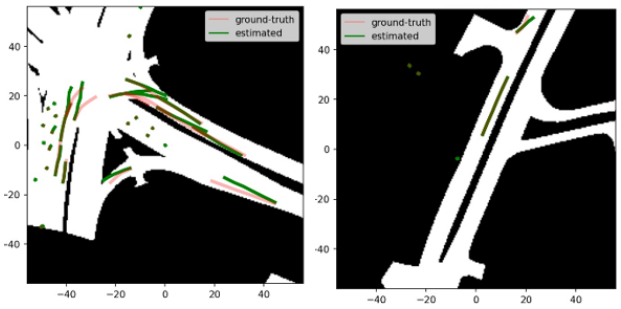

Figure 10: The Weakness of our models

Figure 10 shows that the model is vulnerable to agent rotation. In fact, it is believed that agents went straight by only reflecting image context, although agents had to turn around. These problems are found in all models in present papers. The reason for this is that the nuScenes dataset is based on actual driving vehicles. Therefore, 95 percents of the agents in the data only go straight. Such biased data seem to have produced these results and learning by adding a variety of data will solve this problem.

7. Contributions

Hoonhee Cho : He created a **custom data set** using **nuScences-dev-kit** and implemented **Nuscenes.py** code

and **data loader** function. In addition, he implemented a **data augmentation** code so that multiple agents can be used from one image data. Based on the features encoded in past path, he implemented the **agent interaction attention** code for interaction between agents. Also, implemented the **visual attention** code between the agents features and image features so that the model can understand the context of the image. And, implemented **self-attention** functions for each attention modules. He wrote the **main.py** function so that the functions could work correctly, and proceeded to **organize the code** so that all functions could be written in one framework. He proposed a **model architecture** and implemented a **discriminator** that compares Predicted-Path and Ground-truth, the direction of further research. **PPT** is completed together.

Saeyoon Oh : He implemented the **test code**, which is the process of properly recalling the data required for testing from the trained model, and the **Visualization** code, which makes results into an image. In addition, he implemented the **loss function** of the model using the difference value between the displacements in addition to the displacement. He also found the **best-performing model** on CNN used to extract image features(ex. ResNet, MobileNet). **PPT** is completed together.

Kunho Kim : He implemented the **LSTM encoder** function that produces features from past path and the **LSTM decoder** function that produces output path from final encoded features. In addition, he implemented the **loss function** of the model using the difference value between the displacements in addition to the displacement. He also found the **best-performing model** on CNN used to extract image features(ex. ResNet, MobileNet). **PPT** is completed together.

Hyojin Sim : She understood and implemented ADE and FDE functions, which are typical methods of comparing the performance of Path trajectory. Thanks to her, our models could be measured the performance **numerically**. And, implemented **self-attention** functions for each attention modules. She also found the **best-performing model** on CNN used to extract image features(ex. ResNet, MobileNet). **PPT** is completed together.

References

[1] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom. (2019), *nuScenes: A multimodal dataset for autonomous driving.*, arXiv preprint arXiv:1903.11027.

[2] Cui, Henggang et al. "Multimodal Trajectory Predictions for Autonomous Driving Using Deep Convolutional Networks." 2019 International Conference on Robotics and Automation (ICRA) (2019): n. pag. Crossref. Web.

[3] T.Phan-Minh et al. "CoverNet: Multimodal Behavior Prediction using Trajectory Sets". arXiv:1911.10298 [cs.LG]

[4] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei,and S. Savarese. Social LSTM: Human trajectory prediction in crowded spaces. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016

[5] Nicholas Rhinehart, Kris M. Kitani, and Paul Vernaza. R2P2:A reparameterized pushforward policy for diverse, precise generative path forecasting. In The European Conference on Computer Vision (ECCV), September 2018

[6] Zhao, T., Xu, Y., Monfort, M., Choi, W., Baker, C., Zhao, Y., Wang, Y., Wu, Y.N.: Multi-agent tensor fusion for contextual trajectory prediction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 12126–12134 (2019)