Multimodal Autonomous Vehicle Trajectory Prediction

Andrew Plesniak [1] **Chang Shi** [2] **Dai Li** [2] **Tiancheng Zhao** [3] **Zhaocheng Du** [4]

Abstract

Trajectory prediction is a hot topic in autonomous driving and plays a key role in the success of autonomous driving. During the exploration on existing models, we found that temporal information is essential for long-term prediction, and better representation and better modeling of interaction between agents and between agent and environment can substantially improve the performance. Therefore, we plan to fuse lidar data to get better representation, explicitly model temporal information of all agents, and model interactions between agents and between agent and environment with the minimal computational cost.

1. Code Repository

https://github.com/ChangShiRaine/Multimodal-Navigation

2. Introduction

Autonomous driving depends on the vehicle's ability to predict the trajectories of multiple agents in a scene. Trajectory prediction is the problem of predicting the spatial coordinates of various agents that demonstrate various dynamic behaviors. A well-designed model for trajectory prediction should be able to predict the trajectories of multiple agents in the scene, which are reasonable and consistent with the scene semantics. The computational cost is also important when considering real-time applications.

[1]Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA [2]The Robotics Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA [3]School of Architecture, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA [4]Department of Civil and Environmental Engineering, Carnegie Mellon University, Pittsburgh, Pennsylvania, USA. Correspondence to: Andrew Plesniak <aplesnia@andrew.cmu.edu>, Chang Shi <changshi@andrew.cmu.edu>, Dai Li <dail@andrew.cmu.edu>, Tiancheng Zhao <tianchen@andrew.cmu.edu>, Zhaocheng Du <zhaoched@andrew.cmu.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020.

Currently, most models focus on short-term prediction of agent's trajectory while paying few attention to long-term prediction and they tend to only predict the target agent's trajectory instead of all agents' trajectories in the scene. Also, most models cannot handle complex situations, such as crossroads and crowd road situation, well. Last but not least, most models suffer from substantial computational cost, especially when the number of agents in the scene is large and thus can hardly be applied to real-time applications.

In this paper, we plan to fuse lidar data and map data with methods, like bilinear pooling and corss-modality attention, to get a better spatial representation and hope it can further enhance performance. Also, we plan to explicitly model the temporal information of all agents (instead of only the target agent) with encoder-decoder structure, which is probably transformer, as it allows parallelization for acceleration and tend to achieve better results than LSTM. Last but not least, we plan to enforces the predicted trajectory to lie within the drivable area.

Currently, we implemented three baseline models, MTP, CoverNet and Trajectron++, and hope to get a better understanding of failure cases. We find that our hypothesis matches the experiment results and thus we will explore our research ideas in the next phase of this project.

3. Related Work

Many traditional methods are based on the vehicle dynamic. However, the limitation of these models are that they cannot capture the complex interactions between the target vehicles and environment/other agents. With outstanding performance in many complex tasks, deep learning methods is introduced into this work. Based on our research, the recent works in the deep learning based trajectory prediction model can be roughly divided into several groups.

3.1. Naive Method

We call this group of models "naive" is because (1) they didn't explicitly model the interaction with additional technique with methods like GNN and attention mechanism and (2) they didn't encode the history trajectory with LSTM based model. Most of them apply an end to end methods

for trajectory prediction. For example, the MTP model (Cui et al., 2019) encodes the multimodal information of the scene in a rasterized image. Conditioned on the image and the targets current state, MTP generates a fixed number of trajectories (modes) and their related probabilities. The loss function they use is a weighted sum of regression and classification. The MultiPath (Chai et al., 2019) and CoverNet (Phan-Minh et al., 2020) model used the similar rasterized image as the MTP. But unlike MTP, MultiPath uses fixed anchors obtained from the train set with unsupervised learning to represent the modes, and outputs residuals with respect to anchors in its regression heads. While CoverNet formulates multimodal trajectory prediction purely as a classification problem and predicts the likelihood of a fixed trajectory set. With this setting, CoverNet is able to reduce the prediction uncertainty. The source code for MTP and CoverNet can be found in the NuScene development toolkits.

3.2. LSTM Based Method

The LSTM Based Method encodes the history trajectory with LSTM encoder. MATF (Zhao et al., 2019) encodes each agents past trajectories and the scene information with LSTM and then uses convolution layers to identify inter-agent dynamics. Then, LSTM are used again in order to decode into trajectories for each agent. Comparing to MATF's convolutions layers, MFP (Tang & Salakhutdinov, 2019) also uses LSTM for encoding but adopts a dynamic attention module to capture both the relationships between agents and the scene context. It shows better capability for modeling interaction comparing to CNN. At last, PRECOG (Rhinehart et al., 2019) is different from above two model. It is a goal-conditioned factorized flow-based generative model that uses likelihood inference to forecast the joint state of all agents. It reasons probabilistically about plausible future interactions between agents given observations of the environment. Latent variables are used to capture the uncertainty in other agents' decisions and factorized latent variables can model decoupled agent decisions even though agent dynamics are coupled. The source code for above three models are available.

3.3. Graph Based Method

The Graph Based Method models the interaction between different agents or lane feature with GNN. The most recent STOA model Trajectron++ (Salzmann et al.) uses a graph-structured RNN to predict the agents interactions and trajectories while considering agent motions and heterogeneous scene data. The LaneGCN (Liang et al., 2020) uses GNN to learn lane graph representations and performs a complete set of actor-map interactions. Comparing to the rasterized map used in MTP (Cui et al., 2019), LaneGCN constructs a lanegraph from vectorized map data and extract map topology features. This representation can effectively capture the complex topology and long range dependencies of the lane graph as well as the complex actor-map interactions. Graph-LSTMs method are also used(Chandra et al., 2020). In this paper, a two-stream graph-LSTM network is used with one stream predicts the spatial coordinates of the future trajectories and another predicts agents' behavior and regularizes the first stream with Dynamic Geometric Graphs (DGG). In addition to that, Spectral Cluster Regularization is used to reduce the error of long-term predictions. The source code of all three models mentioned above are available.

3.4. Attention Based Method

The attention based method tries to directly link the agent's trajectory to the most relevant context in order to achieve good performance. The DATF model (Park et al., 2020) features a Cross-Agent Attention module featuring LSTMs as well as a Agent-to-Scene Attention module built of of CNNs. This paper has source code available. The MHA-JAM model (Messaoud et al., 2020) uses a multi-head attention layer to capture the potential interaction between the target and context (Including the interaction map and trajectories). The feature vector outputs by the multi-head attention layer is used as the input for a LSTM decoder. This paper has no open source code. The WIMP model (Khandelwal et al., 2020) is a recurrent graph-based attention framework that adopts a road network attention module and a dynamic interaction graph to capture interpretable geometric and social relationships. It facilitates joint multi-modal prediction of future states over an arbitrary number of actors within a scene by leveraging information from historical, social, and geometric sources. The source code is available. The UST model (He et al., 2020) effectively models the interlaced influence from both spatial and temporal context by treating time and space dimensions equally to model spatio-temporal context. It integrates 2D locations and discrete time space into one unified 3D space, then learn the spatio-temporal context end-to-end. Besides, it can adapt the data by automatically partitioning the spatio-temporal space. The source code is not available. **Probabilistic Multi-modal Trajectory Prediction with Lane Attention for Autonomous Vehicles**(Luo et al., 2020) is a conventional LSTM encoder-decoder framework integrating instance-aware lane representation and goal-oriented lane attention module to generate diverse predictions of future trajectories. The source code is not available. The TNT model (Zhao et al., 2020) first predicts an agent's potential target states T steps into the future, by encoding its interactions with the environment and the other agents. Then it generates trajectory state sequences conditioned on targets. Finally, it estimates trajectory likelihoods and selects a final compact set of trajectory predictions. The source code is not available.

4. Experimental Setup

4.1. Datasets

We are planning to use nuScenes as our dataset.

The NuScenes dataset is a large-scale dataset for autonomous driving collected in Boston and Singapore. The dataset includes camera images, Lidar sweeps, Radar sweeps, object bounding boxes, map data and sensor data. It comprises 1000 scenes, each of which is a 20 second record. Each scene includes agent detection boxes and tracks hand-annotated at 2 Hz, as well as high definition maps of the scenes. As for dataset split, since the labels for test set are not publicly available, we split the original train set into train set and validation set and use the original validation set as test set.

4.2. Modalities

We plan to use four modalities in NuScenes dataset: camera images, Lidar, maps and history trajectories. We will take episodes for 5 seconds long including 3 seconds of history and 2 seconds of future. Figure 1 shows an example visualization of data in Nuscenes.

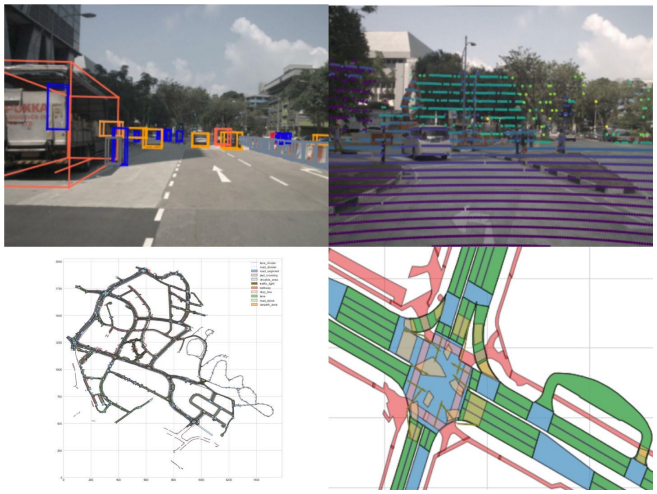

Figure 1. Example visualization of camera images (top left), Lidar (top right), map data (global view on bottom left, local view on bottom right)

4.3. Evaluation Metrics

We will be using the following metrics to evalutate the precision of our prediction.

- **Average Displacement Error over k** (MinADE$_k$): The average of pointwise L2 distances between the predicted trajectory and ground truth over the k most likely predictions.
- **Minimum Final Displacement Error over k** (MinFDE$_k$): The final displacement error (FDE) is the L2 distance between the final points of the prediction and ground truth. We take the minimum FDE over the k most likely predictions and average over all agents.

5. Problem statement

We formulate this problem as a multi-view problem and view trajectory data, which contains temporal information, as one modality and the map data and lidar data (will be integrated later), which contains spatial information, as the other modality. We denote the future states of the target agent i (i.e. the vehicle whose trajectory we are predicting) as X^i,

$$X^i = \{x_t^i, x_{t+1}^i, \ldots, x_{t+H}^i\}_{i=1}^N$$

Here x_t^i denotes the states of target vehicle i at time t, H denotes the prediction horizon. The history states of the target agent i is denoted as S_i, The groundtruth future trajectory is denoted as Y. Then we can formulate the problem of trajectory prediction as calculating the conditional distribution P($\hat{X}$|E, I, S) where E represents the environment (e.g. maps info, traffic lights, traffic signs) and I represents the interaction with surrounding agents (e.g. cars, pedestrians). Our objective function becomes

$$\max_\theta \sum_{i=1}^N E_{S_i} \log p_\theta \left(Y_i \mid E, \mathrm{I}, S\right)$$

Where θ denotes network parameters.

6. Multimodal Baseline models

Describe mathematically 3 multimodal baseline model for your research problem. Your mathematical description should include at least the loss function. Describe in text how the model is optimized and how inference is performed.(About 1 page)

6.1. MTP

MTP first rasterizes an agent-specific to encode the agent's map surrounding and neighboring agents, such as other vehicles and pedestrians. Then, given i-th agent's raster image and state s_{ij} at time step t_j, MTP uses a CNN model to predict a multitude of M possible future state sequences $\{[s_{im(j+1)}, \cdots, s_{im(j+H)}]\}_{m=1,\cdots,M}$, as well as each sequence's probability p_{im} such that $\sum_m p_{im} = 1$, where m indicates mode index and H denotes the number of future consecutive time steps (or prediction horizon). Without the loss of generality, MTP infers agent's future x, y positions instead of full states, while the remaining states can be

derived by the future position estimates. Both past and future positions at time t_j are represented in the agent-centric coordinate system derived from agent's state at time t_j.

MTP takes an agent-centric RGB raster image and agent's current state (velocity, acceleration, and heading change rate) as input, and outputs M modes of future x, y positions ($2H$ outputs per mode) along with their probabilities (one scalar per mode). This results in $(2H + 1)M$ outputs per agent. Probability outputs are passed through a $softmax$ layer to ensure they sum to 1.

MTP defines a single-mode loss L of the i-th agent's m-th mode at time t_j as average displacement error (or l_2-norm) between the points of ground-truth trajectory y_{ij} and predicted trajectory of the m-th mode $\tilde{y}_{imj}$,

$$L(y_{ij}, \tilde{y}_{imj}) = \frac{1}{H} \sum_{h=1}^{H} \|y_{ij}^h - \tilde{y}_{imj}^h\|_2$$

where y_{ij}^h and $\tilde{y}_{imj}^h$ are 2-D vectors representing x, y positions at horizon h of y_{ij} and $\tilde{y}_{imj}$,respectively.

In order to prevent multiple predicted trajectories from collapsing into single trajectory, MTP first runs the forward pass of the neural network to obtain M output trajectories. Then it identifies mode m^* that is closest to the ground-truth trajectory according to an arbitrary trajectory distance function $dist(y_{ij}, \tilde{y}_{imj})$. Additionally, MTP forces the probability of mode m^* to be as close as possible to 1 with $crossentropy$ loss. The final loss function can be written as,

$$L_{ij}^{MTP} = -\sum_{m=1}^{M} I_{m=m^*} \log p_{im} + \alpha \sum_{m=1}^{M} I_{m=m^*} L(y_{ij}, \tilde{y}_{imj})$$

where I_c is a binary indicator function equal to 1 if the condition c is true and 0 otherwise.

6.2. CoverNet

Comparing to other methods, CoverNet simplifies the trajectory prediction problem by instead frame it as classification over a diverse set of trajectories. The author structured the trajectory set to a) ensure a desired level of coverage of the state space, and b) eliminate physically impossible trajectories. To further optimize based on the fixed trajectory, dynamically generating trajectory sets based on the agent's current state can further improve our method's efficiency. Since we did our experiments on fixed trajectory sets, we will only briefly summarize how CoverNet works on the fixed trajectory sets.

A trajectory set is considered to be fixed if the trajectories that it contains do not change as a function of the agent's current dynamic state or environment. Intuitively, this makes it easy to classify over since it allows for a fixed enumeration over the set, but may result in many trajectories that are poor matches for the current situation. Given a set of representative trajectory data, the problem of finding the smallest fixed approximating trajectory set $\mathcal{K}$ can be cast as an instance of the NP-hard set cover problem. In CoverNet, the author uses a coverage metric δ defined as the maximum point-wise Euclidean distance between trajectories. The trajectory set construction procedure starts with subsampling a reasonably large set $\mathcal{K}'$ of trajectories (Here specifically size 20,000) from the training set. Selecting an acceptable error tolerance ε, the author proceed to find the solution to:

$$\begin{array}{ll} argmin_{\mathcal{K}} & |\mathcal{K}| \\ subject\ to & \mathcal{K} \subseteq \mathcal{K}' \\ & \forall k \in \mathcal{K}', \exists l \in \mathcal{K}, \delta(k, l) \leq \varepsilon \end{array}$$

where $\delta\left(s_{t:t+H}, \hat{s}_{t:t+H}\right) := \max_{y=t}^{t+H} \|s_y - \hat{s}_y\|_2$.

By employ a simple greedy approximation algorithm to solve the problem above, the author cherry-pick the best among candidate trajectories to place in a bag of trajectories that will be used as the covering set, then repeatedly consider as candidates those trajectories that have not yet been covered and choose the one that covers the most uncovered trajectories (ties are broken arbitrarily).

After generating the trajectory set, the simple classification network using theconcatenation of convolutional result of local scene raster input and the state inputs(speed, accelaration and yaw) as input is used to output the probability over this set. For the classification losses, cross-entropy is utilized with positive samples determined by the element in the trajectory set closest to the actual ground truth in minimum average of point-wise Euclidean distances.

6.3. Trajectron++

Trajectron++ is a graph-structured recurrent model that takes into agent to agent interaction through its graph structure as well as agent to environment interaction through its map encoder. In order to create the graph, the scenes agents are instantiated as nodes and directed edges are placed based on agent proximity. The nodes contains the semantic agent type (vehicle, pedestrian, bus, etc...) as well as the trajectory sequences.

In its simplest form, Trajectron++ is a form of a Conditional Variational Auto Encoder (CVAE). It uses a number of encoders to end up with a latent vector that encodes all of the scene information. These encoders include: node history encoder which encodes the previous trajectory of the agent, the edge encoder which encodes the edge features of all the neighboring nodes of the same semantic type and then performs attention across the different semantic types, and the map encoder which encoders the semantic map information. The node history encoder and the edge

encoders are both LSTM based where as the map encoder is a CNN. Optionally, there is a robotic future encoder which encodes the future trajectory of the agent being analyzed, which allows the model to condition on what the agent's intention which is often available in real-world system as an output of a motion planning algorithm. Finally during training, the future trajectories of all the other nodes are also encoded and incorporate into the loss function to promote faster learning. The outputs of all these encoders are concatenated and passed through a fully connected layer to produce a latent vector encoding the scene.

On the decoding side, the latent vector is passed through a fully connected layer and GRU decoder. The output is the GRU are bi-variate Gaussian distributions parameters that can be sampled from to get control instructions (acceleration and steering rate) for the dynamics models defined for each semantic class. For example, pedestrians are modeled as single integrators and vehicles as dynamically-extended unicycles. The advantage of this approach as opposed to directly predicting the agents trajectory points is that the distribution control output format allows for a distribution of trajectories rather than a single one. In addition, the dynamic models ensure that the trajectories are dynamically feasible by incorporating dynamic constraints and plausible movement definitions.

The model is trained by maximizing an ELBO function adapted from InfoVAE to account for the conditional formulation.

$$\max_{\phi,\theta,\psi} \sum_{i=1}^{N} \mathbb{E}_{z\sim q_\phi(\cdot|\mathbf{x}_i,\mathbf{y}_i)}\big[\log p_\psi(\mathbf{y}_i \mid \mathbf{x}_i, z)\big] - \beta D_{KL}\big(q_\phi(z \mid \mathbf{x}_i, \mathbf{y}_i) \parallel p_\theta(z \mid \mathbf{x}_i)\big) + \alpha I_q(\mathbf{x}; z),$$

7. Experiment methodology

The experiments of the three baselines are all done on the Nuscenes dataset, while with a little difference on the training/testing split scheme. We all take 3s as the prediction horizon.

For MTP, all the experiments are done on the whole Nuscenes dataset, using the given splits from the benchmark(700 train, 150 val). As the original paper mentions that predicting 3 different trajectories offers the best result and adopting the angle between the current forward direction and the last point of the predicted trajectory as distance metric performs better when changing directions (turn left or turn right), we follow their recommendations. Additionally, we adopt try predicting 2 different trajectories and predicting 16 different trajectories. We use ADE@3s, FDE@3s as the evaluation metric. We adopt ResNet-50 as the backbone to extract features from the BEV raster images.

The CoverNet is trained and tested on the part 1 subset of the Nuscenes dataset(85 train and 15 val), leading to a bit more prediction error than the original paper results. Also, the CoverNet implementation offers several choice of backbone, ResNet-50 and MobileNet_v2, while the original paper run the experiments with ResNet-50, we run the experiments with MobileNet_v2. We take future horizon of 3s, batch size of 16. And since CoverNet define the prediction as a classification problem over a set of pre-computed trajectories(modes) within the kinetics constraints. We tested with 64, 415 and 2206 number of modes, but all over fixed trajectory set coverage, since the authors didn't release the dynamic trajectory set. We use point-wise Euclidean distance as distance metric, and evaluated final results with metrics minADE_1, minADE_5, minADE_{10} and FDE (FDE if calculated over the one most probable trajectory).

For trajectron++, all the experiments are done on the whole Nuscenes dataset, using the given splits from the benchmark(1000 scenes in total, 700 train, 150 val, and 150 test). There is 1 base model and its 3 variations (Dynamics Integration, Dynamics Integration + Maps, Dynamics Integration + Maps + Robot Future). Since we don't use Robot Future information, we don't include results of the last variation in this report. We show FDE and ADE results.

8. Result and Discussion

8.1. Quantitative results

The experiment results of MTP is shown as 1. For MTP, the more trajectories it predicts, the larger the error between the closest trajectory and the ground truth. The experiment results of CoverNet is shown as 2. As we can see that the accuracy improves with using more fine-grained defined trajectory set. The experiment results of Trajctron++ is shown as 3. As we can see, afte adding dynamic integration, maps and future information, the performance boosts.

The MTP performs better than the CoverNet, which is contrary to the original paper of CoverNet. One reason may be that MTP is trained on the whole Nusecenes dataset, while the CoverNet is trained on subset1 of the NuScenes dataset. The other reason maybe that MTP uses ResNet-50 as backbone, while CoverNet uses MobileNet_v2, which performs worse in feature extraction from raster images than ResNet-50 does. The Trajectron++ performs better than the MTP and the CoverNet, which corresponds to the original paper. The reasons include Trajectron++ considers and models historical agents' trajectories, it also explicitly models the agent-to-agent interaction and the agent-to-environment interaction, and it considers the spatial-temporal information as well.

After all, from the experiments, we find that the amount of data used for training, the quality of the map representation, the spatio-temporal information and the interaction between agents and between agents and environment all play a key role in trajectory prediction.

8.2. Error Analysis

After analyzing the predictions from MTP and CoverNet, we find that they perform well on predicting stright trajectories while performing badly on prediction trajectories involve turns. Also, MTP and CoverNet performs well on short-term predictions while performing badly on long-term predictions. We think the reason is that these two models don't consider and model temporal information.

In order to further identify possible weaknesses, we qualitatively analyzed some of the Tracjectron++ predictions. In the figures below, the white dashed line represents the ground truth trajectories while the colored point cloud distributions represent the predicted trajectory distributions.The vehicle agents have vehicle icons and the remaining unlabeled trajectories belong to pedestrians or other semantic classes.

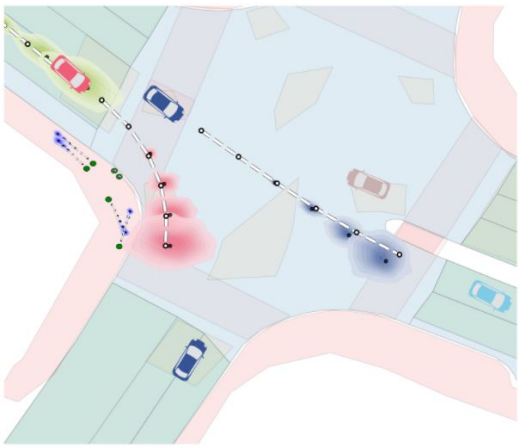

Figure 2. Trajectory example with many agents and complex map information at an intersection.

Looking at some of the visualizations, we gained some interesting insights. Surprisingly, there were some fairly simple situations, such as driving along a straight two lane road with minimal other agents, (see Figures 8 and 9) where the model seemed to incorrectly predict curving trajectories. Whereas in Figure 7, a much more complicated traffic scene with many agents at an intersection the model seemed to perform well. We hypothesize this is due to the graph structure of Trajectron++ that heavily models agent-to-agent interactions. In scenes with few agents and simple semantic information, this may adversely effect the predictions, even though these scenes would seemingly be easier to navigate.

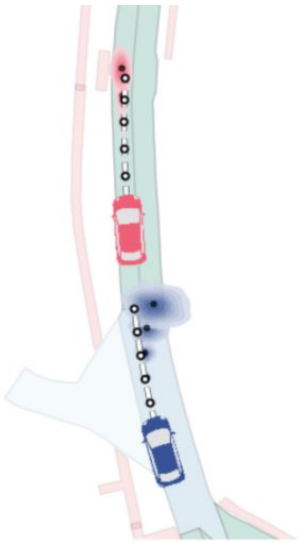

Figure 3. Trajectory example with only two agents on a two lane road.

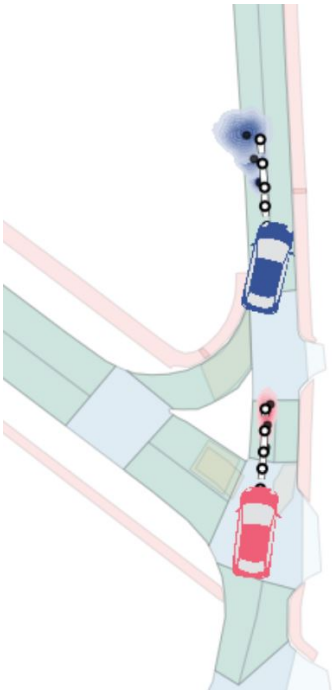

Figure 4. Another trajectory example with only two agents on a two lane road.

Besides, there are some cases where the predicted direction is correct but the predicted vehicle is faster than groundtruth especially when passing crossroads. We argue that it's because when human pass certain locations like crossroads, they tend to slow down to keep safe. But the network fails to learn this.

Drawing from these observations, it may be useful to in-

Method	Modes	minADE_1 @3.0s	minFDE_1 @3.0s
MTP, angle	2	1.6818	3.4305
MTP, angle	3	1.6945	3.4685
MTP, angle	16	1.9213	3.9049

Table 1. Experiment results of MTP on NuScenes dataset

Method	Modes	minADE_1	minADE_5	minADE_{10}	FDE
CoverNet, fixed, $\epsilon = 8$	64	20.2653	0.8979	0.8979	31.2407
CoverNet, fixed, $\epsilon = 4$	415	7.4208	1.7699	1.4739	12.9996
CoverNet, fixed, $\epsilon = 2$	2206	8.0844	6.0647	1.0399	13.2058

Table 2. Experiment results of Covernet (Phan-Minh et al., 2020) on NuScenes Part1 dataset

corporate more environmental information in the form of additional modalities (such as images and Lidar) to bolster situations with lacking map data. In addition, it may also be useful to come up with a way to dynamically balance the contributions of the agent-to-agent interactions vs the agent-to-environment interactions. As a result, the model would focus more of on the environmental information in situations like Figure 8 and 9 where there are few agents. This could possibly take the form of an attention module.

9. Research ideas

We will mainly focus on extension based on the work of (Park et al., 2020) and (Salzmann et al.). Some of the research ideas are listed below. We will compare our final work with prior work from diverse facets: the precision of the predicted trajectory (based on the metrics mentioned in 4.3), performance on extreme corner cases and the multimodal property of the model.

Based on the previous error analysis of current baselines, we conclude that more environmental information are needed and better ways of modeling the interactions between the environment-agent and agent-agent.

(1) Merge more information in the map representations. In the experiments of trajectron++, we notice that merging more information will improve the performance (Dynamic Integration + maps outperforms baseline and Dynamic Integration only). Adding the map information reduces the probability of knocking into the wall and across lane boundaries. So we plan to merge more information in the map representations instead of just using drivable map masks. For example, as shown in 5, we can represent the lane into a lane graph and use a graph convolutional network to encode the lane information and then concatenate it to map features.

(2) Bilinear fusion of images and LiDAR. As what is mentioned previously, we found that the model fails to learn to slow down at crossings and tend to rely heavily on agent-agent interactions. To alleviate this problem, we decide to use camera images and LiDAR modalities since camera images can offer us information like traffic lights, traffic signs and car turning lights while LiDAR can provide more acccurate depth information. In our design as shown in 6, LiDAR and image features are extracted seperately using CNN and fused together by bilinear fusion.

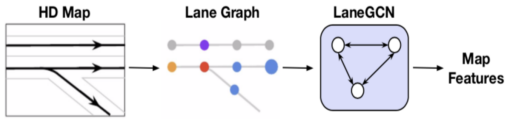

Figure 5. Proposed idea 1

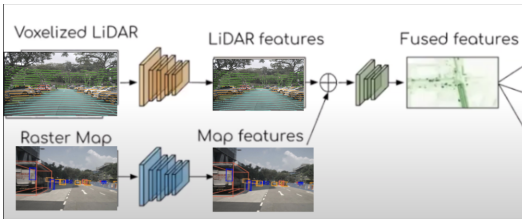

Figure 6. Proposed idea 2

(3) Cross-modality attention. To better align multi-modal modality feature representations and learn to balance agent-environment and agent-agent interactions, we propose to use cross-modality attention. The design is shown in 7

(4) Spatial-temporal attention. Since both MTP and CoverNet don't model spatial information as well as temporal information, they perform worse than Trajectron++. Thus, we believe spatial information and temporal in-

Trajectron++ variants	FDE @3.0s	ADE @3.0s
Base	1.25	0.56
+ Dynamic Integration	1.09	0.44
+ Dynamic Integration, Maps	0.81	0.45

Table 3. Experiment results of Trajectron++ (Salzmann et al.) on NuScenes dataset

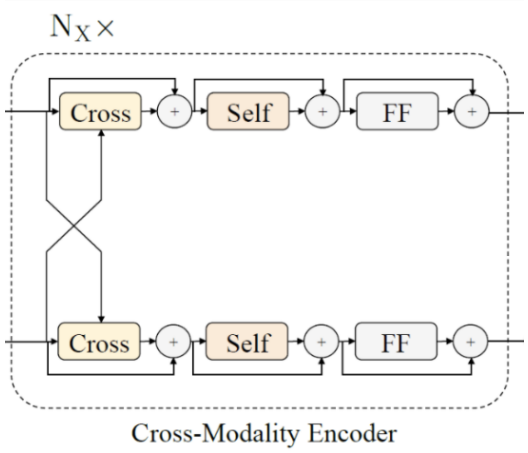

Figure 7. Proposed idea 3

formation play a key role in trajectory prediction. However, Trajectron++ suffers from high computational cost and multi-stage processing. Therefore, we propose to try spatio-temporal attention to reduce computational cost and facilitate end-to-end processing. We think this will further improve performance.

(5) Though a continuous predictive distribution of ectory predictions relying on the ground truth, while (Park et al., 2020) proposed a notation-free approach to estimate the true trajectory distribution based on a drivable-area map. However, it assumes that every drivable location is equally probable for future trajectories to appear in, which is not the real case since human drivers tend not to drive on the edge of the road. Thus we can **merge the ground truth trajectory with the drivable-area map to get a probabilistic drivable-area map** which may better help the trajectory prediction.

(6) The prediction result from most recent models are usually a probabilistic map over all possible trajectories, as shown in 8, neglecting the possible intention of the agents. Apparently, the trajectory would be different when the agent tends to go straight comparing to when the agent tends to turn left. Thus, if we can **predict which class the intention of the agent falls into, and only forecast the trajectory distribution under that intention**, the forecasting accuracy may have an improvement. Possible methods include an embedded intention module with an end-to-end training style as (Zhao et al., 2020) suggested.

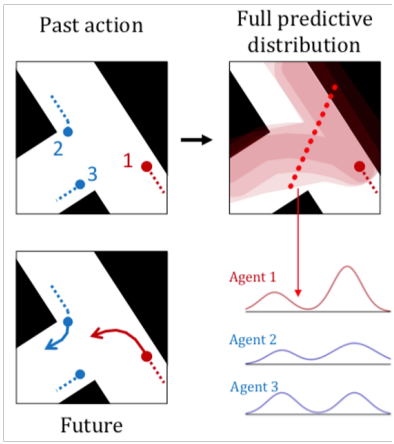

Figure 8. Example of trajectory prediction from (Park et al., 2020)

References

Chai, Y., Sapp, B., Bansal, M., and Anguelov, D. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In *CoRL*, 2019.

Chandra, R., Guan, T., Panuganti, S., Mittal, T., Bhattacharya, U., Bera, A., and Manocha, D. Forecasting trajectory and behavior of road-agents using spectral clustering in graph-lstms. *IEEE Robotics and Automation Letters*, 5(3):4882–4890, 2020.

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

He, H., Dai, H., and Wang, N. Ust: Unifying spatio-temporal context for trajectory prediction in autonomous driving, 2020.

Khandelwal, S., Qi, W., Singh, J., Hartnett, A., and Ramanan, D. What-if motion prediction for autonomous driving, 2020.

Liang, M., Yang, B., Hu, R., Chen, Y., Liao, R., Feng, S., and Urtasun, R. Learning lane graph representations for motion forecasting, 2020.

Luo, C., Sun, L., Dabiri, D., and Yuille, A. Probabilistic multi-modal trajectory prediction with lane attention for autonomous vehicles, 2020.

Messaoud, K., Deo, N., Trivedi, M., and Nashashibi, F. Trajectory prediction for autonomous driving based on multi-head attention with joint agent-map representation. 2020.

Park, S. H., Lee, G., Bhat, M., Seo, J., Kang, M., Francis, J., Jadhav, A. R., Liang, P. P., and Morency, L.-P. Diverse and admissible trajectory forecasting through multimodal context understanding. *arXiv preprint arXiv:2003.03212*, 2020.

Phan-Minh, T., Grigore, E. C., Boulton, F. A., Beijbom, O., and Wolff, E. M. Covernet: Multimodal behavior prediction using trajectory sets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14074–14083, 2020.

Rhinehart, N., McAllister, R., Kitani, K. M., and Levine, S. PRECOG: prediction conditioned on goals in visual multi-agent settings. *CoRR*, abs/1905.01296, 2019. URL http://arxiv.org/abs/1905.01296.

Salzmann, T., Ivanovic, B., Chakravarty, P., and Pavone, M. Trajectron++: Multi-agent generative trajectory forecasting with heterogeneous data for control. *arXiv preprint arXiv:2001.03093*.

Tang, Y. C. and Salakhutdinov, R. Multiple futures prediction, 2019.

Zhao, H., Gao, J., Lan, T., Sun, C., Sapp, B., Varadarajan, B., Shen, Y., Shen, Y., Chai, Y., Schmid, C., Li, C., and Anguelov, D. Tnt: Target-driven trajectory prediction, 2020.

Zhao, T., Xu, Y., Monfort, M., Choi, W., Baker, C., Zhao, Y., Wang, Y., and 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.