Vehicle Motion Prediction Using the Waymo Open Motion Dataset

Project Category: General Machine Learning (Autonomous Vehicles)
Project Mentor: Masha Itkina

Elin Lovisa Byman
Dep. of Electrical Engineering
Stanford University
byman@stanford.edu

Eden Wang
Dep. of Computer Science
Stanford University
eyw@stanford.edu

Emil Vardar
Dep. of Electrical Engineering
Stanford University
eevardar@stanford.edu

Abstract

Motor vehicle accidents are a significant cause of death and autonomous vehicles (AVs) have been suggested as a potential solution in reducing accidental deaths. However, a significant challenge to the adoption of autonomous vehicles remains motion prediction. Our work leverages the Waymo Open Dataset, a collection of scenarios containing "critical scenarios" that demand complex interaction modeling with information on object states and roadgraph features. We develop a two-step trajectory prediction model that first predicts the goals of target objects, then uses these goals as anchors to complete their trajectories. We found our results to affirm the potential of modular approaches in motion prediction tasks, achieving decent performance in goal prediction and high performance in trajectory completion module.

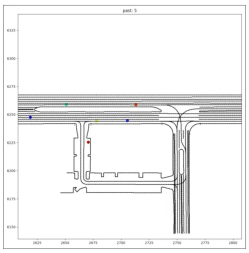
1 Introduction

The World Health Organization estimates that 1.3 million people die from traffic accidents every year [1]. A majority of these accidents are avoidable, a product of human error. With the recent development of autonomous vehicles (AVs), it seems possible to drastically reduce the number of accidents and save lives. In order to do so, and to be able to improve the safety of AVs, one important task that has attracted a lot of attention is to be able to accurately predict the motion of nearby objects. This project aims to predict the trajectories of vehicles, pedestrians and cyclists (referred to as 'objects') up to 8 seconds into the future provided with one second of context. The context includes information about the current scene such as the coordinates of streets and lanes, the states and locations of traffic lights and signs, and the positions, velocities, and directions of all objects in the scene. In our project, we build a tractable model that can tackle the problem of motion prediction in critical situations. To do so, we leverage a modular approach with two steps, first predicting the desired end points for all objects, then estimating the trajectory conditioned on these end points.

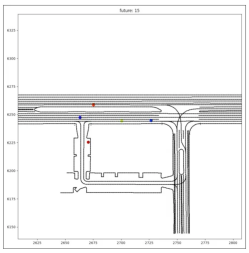
2 Related Work

Motion prediction remains an ongoing challenge in the research community. Open competitions are frequently held encouraging teams to share their best models and iterate on previous approaches. Top scoring teams often publish their work and share their approaches with the broader community. Several projects that have garnered attention in the community include works by Konev et al., Zhao et al., and Gu et al. [2, 3, 4]. Though a rich set of approaches exist, including constant velocity and latent behavior approaches, for our purposes we can divide approaches into end-to-end and modular approaches, with the latter having shown significant potential in recent competition settings.

End-to-end approaches primarily vary in preprocessing, feature use, and model architecture. One of the primarily inspirations for this project is a naïve CNN model that rasterizes the scenario and leverages convolutional layers to generate predictions directly [2]. On the other hand, modular approaches may differ in the intermediate steps taken to perform the overall motion prediction task. For instance, two of the most promising approaches that achieve state-of-the-art performance in recent Waymo motion prediction challenges are the Target-driveN Trajectory (TNT) and the DenseTNT models, both of which first rank plausible endpoints from the map, then construct trajectories conditioned on these endpoints, and finally select the best trajectories [3, 4].



(a) Scene at time step 0.



(b) Scene at time step 20.

Figure 1: Example of one scene at two different time steps.

3 Dataset and Features

Various datasets have previously been curated and used as standards for motion prediction models, including the Lyft Level 5, NuScenes, Argoverse, and Interactions datasets [5, 6, 7, 8]. We used the Waymo Open Dataset, a newer dataset that prioritizes modeling 'critical situations', such as scenes at busy street interactions with many pedestrians or scenes with complex acceleration and braking behaviors [9, 10]. The dataset consists of 103,354 segments, each containing 20 seconds of object detection sampled at 10 Hz. The segments are divided into 9 second long segments with a 5 second overlap, which creates a total of 310,062 segments. Each segment has a corresponding scene containing information about the surroundings such as the roads, lanes, and sidewalks, object states and traffic light states. Furthermore, each segment contains pertinent information on up to 128 different objects in the scenario, including their type, size, position, velocity, yaw, among other features. Partially observed objects are marked as such in the scenario data. One example of one scene at two different time points can be seen in figure 1.

The dataset is split into training, validation and test set, corresponding to 70%, 15% and 15% of the data. The test set is not publicly available, so in this project, the validation set is split in half into a validation and a test set. Our pre-processing step includes a normalization step where features are centered about the origin so as to stabilize training. Feature extraction is one of the main routes of experimentation in our project design; our pre-processing step rasterizes each segment's context and determines what features are most relevant for each model. Specifically in our work, we included most object and roadgraph features, choosing to omit less pertinent features such as object size and sparse roadsign features such as traffic lights and their states.

4 Methods

4.1 Model Overview

Our approach was primarily inspired by the works mentioned in section 2. We combine both the modular approach described in the TNT and DenseTNT works with a simpler convolutional approach that was able to perform on par with more complex, state-of-the-art models. However, unlike the TNT and DenseTNT works, we chose to only output a single goal and trajectory for each object, omitting an additional trajectory ranking module. Thus, our final model consists of two modules: a goal prediction module and a trajectory prediction module. We extract relevant features as described above and vectorize the provided information for use as the input to each model in our pipeline. This approach allows us to identify and select features that are more relevant to a given module, reducing irrelevant information and the complexity of our overall model. The motion prediction task, as specified by the Waymo challenge, evaluates performance by requiring models to predict trajectories for up to eight objects in a given scenario. Therefore, both our goal and trajectory modules generate outputs for eight different objects at a time in a given scenario.

4.2 Long Short-Term Memory

Long Short-Term Memory (LSTM) blocks are usually used in sequential/temporal applications and their structure can be seen in figure 2. Their recurrent structure makes it possible to better learn sequences of data, with the memory component storing pertinent information that can be referenced later in a sequence. Since our model contains 10 discrete states containing past context and 1 discrete state containing current context, which are temporally related, LSTM blocks seem to be a strong approach that can capture these dependencies. As seen in figure 2, the block in the middle does not only take information about the current time step (X_t) but it also retrieves information from all previous steps (X_1, X_2, ..., X_{t-1}). Hence, the outputs at time t (Y_t) depend on all the previous inputs in addition to the current input. For more details on the concepts behind LSTM, we refer to [11, 12].

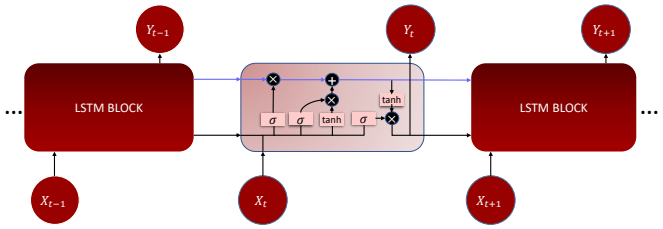


Figure 2: Internal LSTM cell structure.

4.3 Goal Prediction

The principle behind our goal prediction model is to estimate the endpoint of objects 8 seconds in the future provided with 1 second of past context. These estimates will then be used as anchors to predict object trajectories in the subsequent model. Though we introduced the LSTM chain above as having inputs and outputs in every block, we used an alternate structure in our goal prediction and trajectory completion modules. To estimate target positions 8 seconds into the future, we used the model provided in figure 3. The model can be deconstructed into a time-based component and a static component. The time-based LSTM chain sequentially processes inputs containing past states (including object velocity, yaw, and position). The chain contains eleven blocks (ten for the 1 second of past context, one for the current state), and only the output from the last block is used in the final dense layer.

In addition to the time dependent object information, each scene also contains static information such as the underlying roadgraph and other time-independent features. These features do not change over each timestamp and are instead processed independently. Static features are fed through four convolutional layers, each followed by a max pooling layer. To decrease the overfitting, we also add a dropout layer here. The output of the dropout layer is concatenated with the time-dependent outputs from the LSTM block. The resulting flattened vector is then fed into a dense layer and the output of the last dense layer is used as the endpoint predictions for all the vehicles we need to predict.

4.4 Trajectory Completion

The trajectory completion model receives as input a preprocessed scenario and couples it with the estimated endpoint given from the goal prediction model. The goal of this model is to predict 80 coordinates, ten for each second in the future, for each object conditioned on their goal. The first half of this model is similar to the goal prediction model given in figure 3 (the 'encoder'). Similar to the goal prediction model, time-dependent features are processed independently of static features prior to use in the decoder. The only difference between our previous model and the trajectory encoder is that encoder inputs now also contain the broadcasted predicted end points.

To predict the future points, we also define a 'decoder', given in figure 4. The decoder contains 8 LSTM cells, with each cell responsible for predicting one second of future motion. This architecture is similar to the cell structure described in section 4.2 with inputs being 0. The outputs of each LSTM should, when passed through a dense layer, output 10 points for each object, corresponding to one second of the eight-second prediction window. With all eight cells, we obtain the full 8-second predicted trajectories for all target objects.

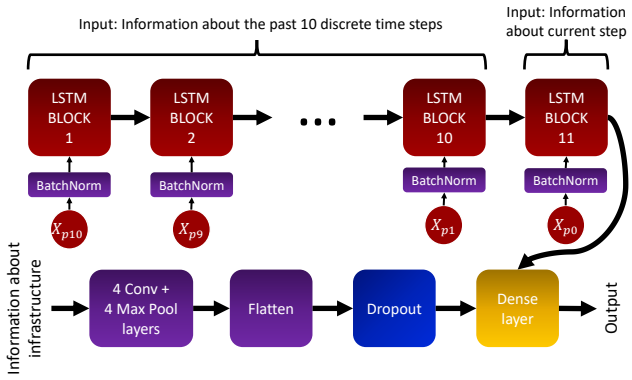


Figure 3: End point prediction model.

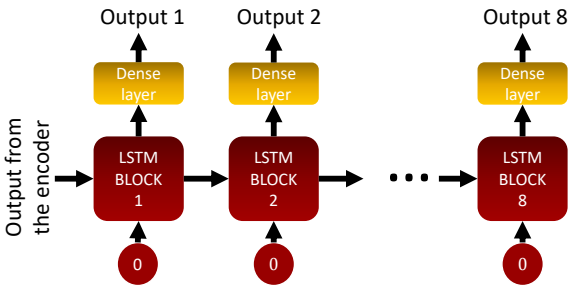


Figure 4: Trajectory prediction model

5 Results and Discussion

5.1 Hyperparameters and Metrics

Hyperparameters for each module were found through a logarithmic range search. The best values found for each modules are specified in table 1. Mean square error was used as the loss function for both modules as it can be interpreted as the squared distance between the ground truth and the predicted coordinates, and the goal is to minimize this distance. For both modules, the loss function was minimized using an Adam optimizer with parameters $\beta_1 = 0.9, \beta_2 = 0.999$. To evaluate model performance, we employ two metrics described in [10], namely Final Displacement Error (FDE) and Average Displacement Error (ADE). The former evaluates the L2 norm between the prediction and the ground truth at the final time stemp, whereas the latter computes the average norm over all timestamps.

Table 1: Hyperparameters for the two modules.

Goal Prediction Module			Trajectory Completion		
Learning Rate	Batch Size	Dropout	Learning Rate	Batch Size	Dropout
0.001	128	0.4	0.001	128	0.4

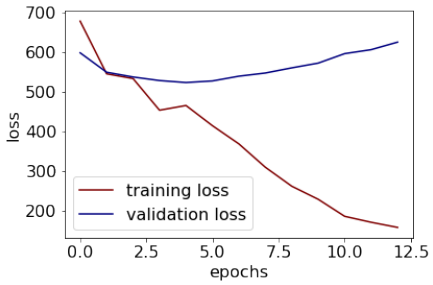
5.2 Goal Prediction

The loss for the training and the validation set for the goal prediction model can be seen in figure 5a, where the best validation loss is 523.20 and the best training loss is 157.48, however the training loss was still declining rapidly when training was halted. This corresponds to an FDE of 22.8 and 12.5 meters in our validation and training sets, respectively. Furthermore, from figure 5a it is clear that the model overfits to the training data. The overfitting can also be seen by comparing the ground truth coordinates with the predictions made on the training and validation set as pictured in figure 6. Here, the predictions on the training set are more accurate than the predictions on the validation set.

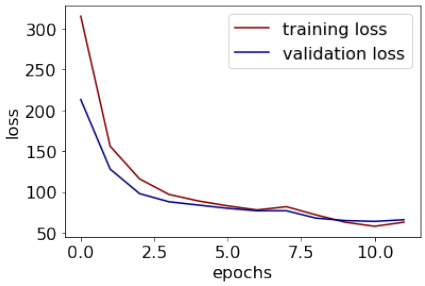
One possible reason for our observed results is the limited regularization used in the model. To reduce variance, dropout could be applied not only to the static information but to the time dependent information as well. The dropout rate for the static information could also be increased in tandem with other regularization techniques. Given our computational and time restraints (with each training epoch taking 45 minutes for a single model), it was unfortunately not possible to make additional changes within the scope of this project. Another approach to reduce variance is to increase the size of the training dataset (e.g., with augmentation techniques).

5.3 Trajectory Completion

The loss for the trajectory completion model is shown in figure 5b. Compared to the goal prediction model, both the training loss and the validation loss display a steeper descent, indicating better model generalization with small variance. This is expected as trajectory completion anchored on an endpoint is easier when compared against predicting goals as far as 8 seconds into the future. Observe that we used the ground truth values as end points in the training phase. The best validation loss for the trajectory completion is 63.90, and the best training loss is 58.29. This corresponds to an ADE of around 8.0 and 7.6 meters on the validation and training set respectively.

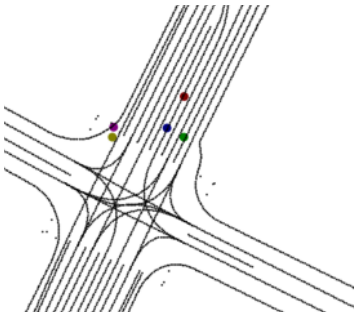


(a) Loss for the goal prediction model.

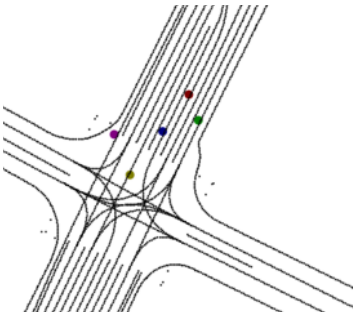


(b) Loss for the trajectory completion model.

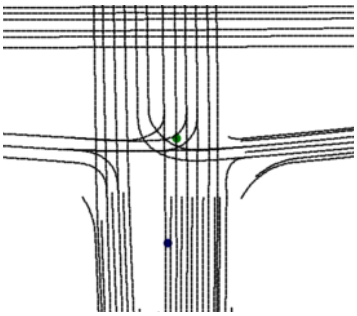
Figure 5: The loss functions during training of the two models.



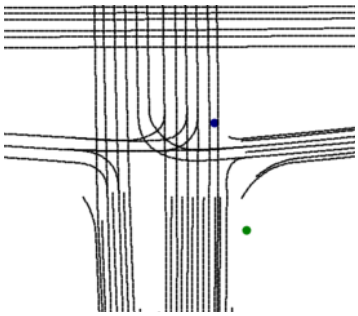
(a) GT for training example



(b) Prediction for training example



(c) GT for validation example.



(d) Prediction for validation example.

Figure 6: The ground truth (GT) positions and the predicted positions for a training and a validation example.

5.4 Combined model

The combined model was evaluated using a withdrawn test set. The endpoints were predicted using the goal prediction model and the rest of the trajectory was predicted using the trajectory completion model. The final displacement error (FDE) was 23.95, and the average displacement error (ADE) was 13.15.

6 Conclusions and Future Work

We partitioned the task of motion prediction into two subtasks: goal prediction and trajectory completion. Our implementation of the goal prediction model had a very high variance and overfitted to the training data, which decreased the overall performance of the system. One way to improve upon on these results is to increase the regularization or the training data to reduce the overfitting of the goal prediction module. Our trajectory completion model had a far better performance and seems promising as a trajectory completion model. However, due to the low performance of the goal prediction model, our overall model's ADE and the FDE (13.15 and 23.95, respectively) are still higher than what would be acceptable in real world applications. Potential avenues for future work include evaluating other model architectures (using 80 LSTM cells, one for each timestamp), further tuning the hyperparameters of the models with a greater focus on the goal prediction model, and evaluating the incorporation of other static and time-dependent context features.

7 Contributions

All team members contributed to literature review, establishing methodology, and hyperparameter tuning. Most of the remaining work was collaborative, but the main responsibilities were divided as follows:

- Emil Vardar: Mainly responsible for the base model architectures for both the goal prediction model and the trajectory completion model.
- Eden Wang: Mainly responsible for the trajectory completion module.
- Elin Lovisa Byman: Mainly responsible for the goal prediction module.

References

[1] World Health Organization. Road traffic injuries. URL: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries, Jun. 21, 2021 [Online].

[2] Stepan Konev, Kirill Brodt, and Artsiom Sanakoyeu. Motioncnn: A strong baseline for motion prediction in autonomous driving. In *Workshop on Autonomous Driving, CVPR*, 2021.

[3] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Benjamin Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. Tnt: Target-driven trajectory prediction. *arXiv preprint arXiv:2008.08294*, 2020.

[4] Junru Gu, Qiao Sun, and Hang Zhao. Densetnt: Waymo open dataset motion prediction challenge 1st place solution. *arXiv preprint arXiv:2106.14160*, 2021.

[5] Lyft. Lyft Level 5. URL: https://level-5.global/data/, 2022 [Online].

[6] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.

[7] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019.

[8] Wei Zhan, Liting Sun, Di Wang, Haojie Shi, Aubrey Clausse, Maximilian Naumann, Julius Kummerle, Hendrik Konigshof, Christoph Stiller, Arnaud de La Fortelle, et al. Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps. *arXiv preprint arXiv:1910.03088*, 2019.

[9] Waymo. Waymo Open Dataset. URL: https://waymo.com/open/download/, 2022 [Online].

[10] Scott Ettinger, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai, Ben Sapp, Charles R Qi, Yin Zhou, et al. Large scale interactive motion forecasting for autonomous driving: The waymo open motion dataset. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9710–9719, 2021.

[11] F.A. Gers and E. Schmidhuber. LSTM recurrent networks learn simple context-free and context-sensitive languages. *IEEE Transactions on Neural Networks*, 12(6):1333–1340, 2001.

[12] Gábor Melis, Tomáš Kočiskỳ, and Phil Blunsom. Mogrifier LSTM. *arXiv preprint arXiv:1909.01792*, 2019.