Motion Prediction of Agents in the Vicinity of Self Driving Car

Vaishnavi Gejji [1] **Amey Deshpande** [1] **Harishkumar Ramadhas** [1] **Harin Vashi** [1]

Abstract

The goal of the project was to predict the motion of an autonomous vehicle and the surrounding agents given their trajectory for the past one second. For this, we used rasterized images as an input to a CNN baseline. To the given parameters, we added the velocities along x and y direction. This helped predicting instantaneous velocity of the agent at every time step. In addition, the baseline was modified by adding an LSTM decoder to study their impact on predictions. Instantaneous velocity was an added parameter for the model and predicting instantaneous velocities can be used to improve the motion prediction of the agents and can facilitate agent interaction and cooperative driving.

1. Introduction

Problem before solution: Autonomous Vehicles will soon be part of our roads and they will share the road with other traffic participants. The Autonomous vehicle needs to estimate the surrounding traffic agents' intentions, motions and human driving intentions to safely plan it's path without colliding to any of the agents. The process of actively anticipating the traffic actor's future behaviour is known as Motion forecasting. Modelling the future behaviour of surrounding agents is a difficult problem and needs to be addressed by the autonomous vehicle to successfully plan it's path. Moreover, there exists uncertainty in the motion of the surrounding agents, the surrounding agent can have uncertain movements which needs to be accounted too. This comes under multi-modal prediction problem (predicting multiple probable trajectories of single agent) which also needs to be addressed. This is an exciting research problem for Autonomous Vehicle domain and Deep learning algorithms are extensively used for predicting future trajectories of the agents. Thus, we leveraged deep learning algorithms like CNN, LSTM to predict the future trajectories of agents by feeding the features (rasterized images of the traffic scene) to the deep learning network. One interesting addition to address the existing problem is that with the prediction of future trajectories, we also predicted the future velocities of the the agent. The future velocity information will be very helpful as the EGO vehicle(the autonomous car) will use it to plan it's own trajectory correctly and generate better interaction models with the surrounding agents.

1.1. Research contributions

In this paper, we propose a novel approach of taking instantaneous velocity of the agents into consideration for improving the motion prediction. All the approaches for this particular problem, to the best of our knowledge, consist of modifying and replacing the baseline with a better performing network.

However, here, in addition to modifying the baseline, we used instantaneous velocity of the agents as an additional parameter for predicting the future trajectories.

2. Related Work

Research in motion and trajectory prediction can be majortly classified into time series models(RNN, LSTM)(1), GAN models, CNN models and Hybrid models. Djuric Et al(3) explored CNN+LSTM architecture, which dealt with uncertainty in the prediction of the motion trajectories. With the above research, we explored Waymo, and Argoverse dataset. As LSTM architecture is vastly used for time series data, we explored the LSTM architecture to see what results we get. We explored Argoverse dataset as it was relatively smaller than Waymo and provided us the in-built APIs to use with the dataset. Moreover, the dataset encoded various features to capture social and spatial context.The dataset has pre-processed features of map and social features of agent trajectory with the timesteps.

Thus, we chose to explore Argoverse dataset.

For Argoverse dataset we explored LSTM encoder-decoder network using different types of combination of features. As LSTM is popularly used for time series data, we chose LSTM encoder- decoder for it. We fed given the past input

[1]Worcester Polytechnic Institute. Correspondence to: Vaishnavi Gejji <vgejji@wpi.edu>, Amey Deshpande <adeshpande@wpi.edu>, Harishkumar Ramadhas <hramadhas@wpi.edu>, Harin Vashi <hvashi@wpi.edu>.

CS/DS 541 Deep Learning Final Project, Worcester Polytechnic Institute.

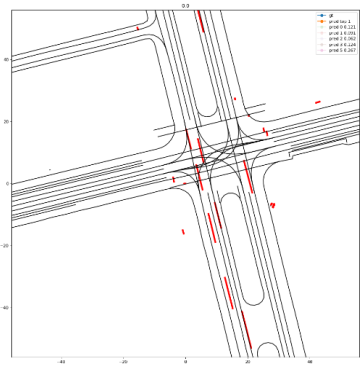

Figure 1: Example of a rasterised image

$$gt = [(x_i, y_i)] \text{ , where } i = 1, 2, \cdots 80$$
$$H_K = [(x_i^k, y_i^k)] \text{ , where } k = 1, 2, \cdots 6$$
$$L = -\log P(gt)$$
$$= -\log \sum_k c^k \prod_t N(gt \mid \mu = H_k, \Sigma = E)$$
$$= -\log \sum_k c^k \prod_t N(x_t \mid \mu = \bar{x}_t^k, \sigma = 1) \cdot N(y_t \mid \mu = \bar{y}_t^{(k)}, \sigma = 1)$$
$$L = -\log \sum_k e^{\log(c^k) - \frac{1}{2}\Sigma_t (\bar{x}_t^{(k)} - x_t)^2 + (\bar{y}_t^{(k)} - y_t)^2}$$

Figure 2: Loss function calculations

coordinates(x,y) of a vehicle trajectory and time steps t = 1, . . . , Tobs to predict the future coordinates (x,y) for time steps t = Tobs+1, . . . , Tpred. We evaluated our predicted trajectories based on the metrics which are minADE, minFDE, mAP.

3. Proposed Method

3.1. Rasterisation

Rasterisation process serves as the dataset pre-processing step for both the models explained in section 3.2 and 3.3. This is a method used to convert raw data into images which can be used for training the network. Pre-rendering the dataset into rasterized format and saving them locally as .npz format files saves considerable amount of space (500GB tfrecord files were converted to 4GB pre-rendered npz files) and training time. Road maps are extracted from the data using this method and the trajectories of the agents are plotted on these maps. There are two main considerations while generating the raster images viz. the velocity of the agent should be along X direction and that the agent under consideration should be at the same location every time. To ensure this, the training images were rotated and shifted during pre-rendering.

3.2. Modifying the baseline - Instantaneous Velocity Prediction

In this approach, the baseline model - MotionCNN was modified to incorporate the instantaneous velocity data of the agent of interest and predict the velocity data for the 80 timestamps over the period of 8 seconds into the future along with the trajectory points for six hypotheses. To predict the velocities of the agents for each hypothesis, we input the instantaneous velocity data along X and Y axis for 11 timestamps which includes 10 past data points and one current data point. Each timestamps is taken in an interval of one-tenth second. The velocities were flattened into a vector and concatenated with the vector output from the Xception-71 CNN module of the MotionCNN architecture. Further multiple blocks consisting of 1-D batch normalization, Leaky Relu and Fully Connected layers with different parameters were added to the architecture Fig:3. Instantaneous Velocity Prediction involves negative values, hence Leaky Relu was used in the newly added layers. The model outputs a vector of size 1926 which then is reshaped into the shape of size 6x(80x2x6)x(80x2x6) [N-Hypotheses x TrajectoryPts x XYVelocities]. For training of this network, negative log likelihood loss function was used along with this, Adam optimizer and a Cosine Annealing scheduler. The loss function calculates the loss for the trajectory and the XY velocities predictions. Then, they their sum is used for backward propagation.

3.3. Modifying the baseline - MotionCNN with LSTM Decoding

The motivation behind implementing a CNN-LSTM model was to maximize the prediction accuracy by exploiting the individual advantages of the two networks. The training data is in the form of images which consist of road maps and positions of different agents within the map. Thus, a CNN is particularly useful in such an implementation, wherein the spatial correlation plays an important role. Thus, this model was used for extracting the features from the training data. Although the number of trainable parameters for this network are relatively more, the ability to accurately predict long time-series data can be useful in motion prediction.

Proposed approach: In this approach we used an LSTM decoder with the Xception71. The features from the last flattened layer of the MotionCNN with the size 2048 is fed to another fully connected layer with the size 64 which was chosen arbitrarily. This feature vector is then used as an

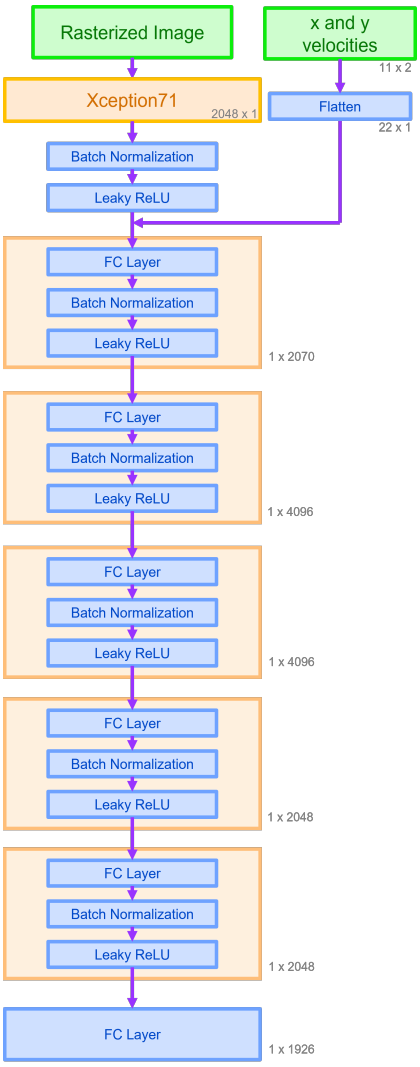

Figure 3: MotionCNN with velocity architecture

input to the LSTM decoder. The cell state is initialized to all zeros. The network used is as shown below:

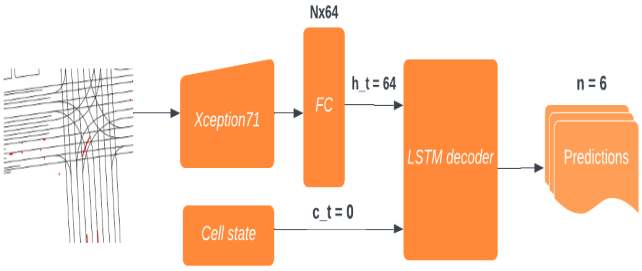

Figure 4: MotionCNN with LSTM decoder

The output of the LSTM is converted into a 2D output using a fully connected layer which consists of predictions for six possible trajectories with the corresponding confidences for each time step.

The outputs of the model after visualization are as follows:

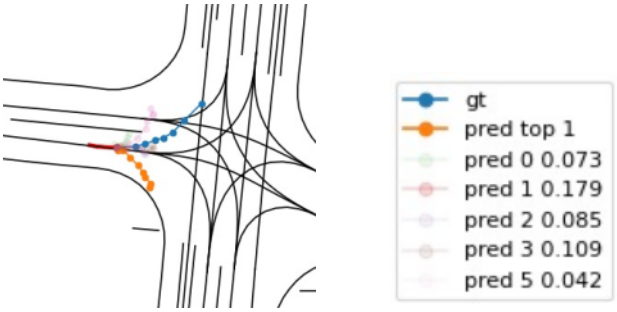

Figure 5: Prediction using baseline(pretrained) model

For training of this network, negative log likelihood loss function was used along with this, Adam optimizer and a Cosine Annealing scheduler.

Limitations on implementation: Although this architecture would theoretically may generate good predictions, due to limited resources and time, it could not be implemented completely.

4. Experiment

The MotionCNN with velocity architecture was compared against the baseline predictions. Both the models were trained on a subset of the WAYMO motion prediction dataset (8) as the entire dataset takes 7 days to train for one epoch in our hardware. We trained it on workstation laptop with Nvidia RTX3070 gpu 8GB VRAM and Intel i7-11800h cpu on ubuntu 20.04LTS. We trained baseline and Velocity architecture with the same subset of the whole training dataset. The training loss for the baseline converged around 125,000 iterations as shown in Fig. 6. How-

ever, our model required more than 250,000k iterations to converge as shown in Fig. 7.

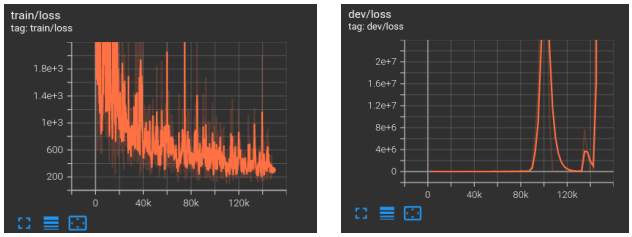

Figure 6: Training and Validation Details of the baseline model

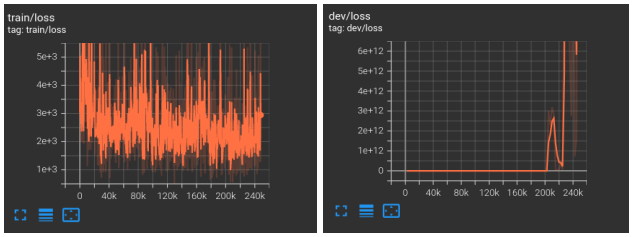

Figure 7: Training and Validation Details of the proposed model

4.1. MotionCNN Baseline

Basic architecture: The baseline architecture was inspired from the MotionCNN paper. It consists of Xception71 as the CNN backbone, followed by a fully connected layer which is used to obtain a total of six trajectories with corresponding confidences.

For this multi-class classification problem, the negative log-likelihood was the loss function used.

In the loss calculations as shown in Fig: 10, i represents the number of time steps to be considered for prediction. As the motion for the next eight seconds is to be predicted with images rendered at 10 frames per second, we need to predict a total of 80 points. Thus,

$$i \in [1, 80]$$

On the other hand, k represents the total number of hypotheses i.e. the number of trajectories to be predicted. In this paper, motion of the agent along six trajectories is to be predicted. Hence,

$$k \in [1, 6]$$

C represents the prediction confidence. x_t and y_t represent the ground truth values for the agent's position coordinates at time step t. Adam optimizer was used along with the Cosine Annealing scheduler for the training of the baseline model.

$$gt = [(x_i, y_i)] \text{ , where } i = 1, 2, \cdots 80$$

$$H_K = [(x_i^k, y_i^k)] \text{ , where } k = 1, 2, \cdots 6$$

$$L = -\log P(gt)$$

$$= -\log \sum_k c^k \prod_t N(gt \mid \mu = H_k, \Sigma = E)$$

$$= -\log \sum_k c^k \prod_t N(x_t \mid \mu = \bar{x}_t^k, \sigma = 1) \cdot N(y_t \mid \mu = \bar{y}_t^{(k)}, \sigma = 1)$$

$$L = -\log \sum_k e^{\log(c^k) - \frac{1}{2}\Sigma_t (\bar{x}_t^{(k)} - x_t)^2 + (\bar{y}_t^{(k)} - y_t)^2}$$

Figure 8: Loss function calculations

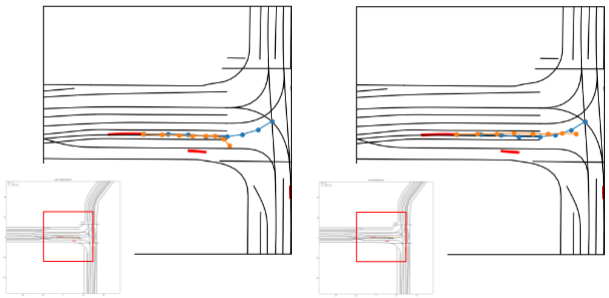

Figure 9: Top 1 Prediction using baseline vs MotionCNN with velocity architecture

5. Results

Metrics	Baseline(Pretrained)	Baseline	VelocityArch.
mAP	0.2136	0.0936	0.0955
minADE	0.7400	0.354	0.242
minFDE	1.4936	0.735	0.682

Table 1: Comparison of metrics values between pretrained baseline model, trained baseline model and MotionCNN with velocity architecture.

For comparing the performance of the proposed model against the baseline (pretrained) and baseline (trained), we considered the minADE, minFDE and mAP metric values (9). The metric values we got after evaluating predicted trajectories with ground truth trajectories are tabulated in the table:1. We have visualized the results of two cases, one where our model performed better than the baseline as shown in Fig: 9 and the other where the baseline model did better as shown in Fig: 10. The velocities were predicted with the Mean Square Error of 4.5m/s.

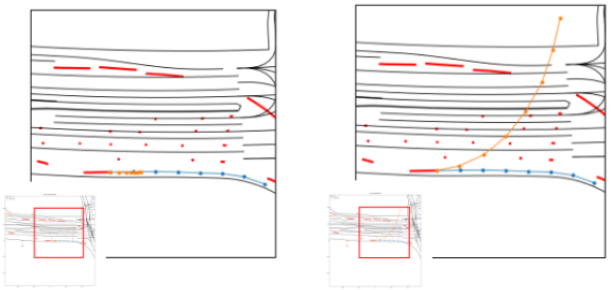

Figure 10: Top 1 Prediction using baseline vs MotionCNN with velocity architecture

6. Discussion

Through this paper we propose that using instantaneous velocity of agents can help in improving the motion prediction results. Also, in the future, this parameter can be used to predict interaction between different agents. Additionally, adding an LSTM decoder to the CNN backbone can help in predicting motion more accurately over longer time periods.

7. Conclusions and Future Work

Both our ideas were materialized into two modified networks. However, we were able to only successfully train and test the MotionCNN - with velocity architecture due to shortage of hardware resources and time. The modified Motion CNN was able to predict the trajectories of agent similar to that of the baseline - Motion CNN model and in cases, it was able to predict better than the baseline model. However, the velocity prediction requires substantial improvement for accurate prediction of the velocity. It fails in few cases with similar effect on output,i.e.,generating motion along a circular path which is way off from the ground truth. This effect needs to be addressed in the future scope.Also, the model can be improved upon by predicting the future velocity from the rasterized images directly as it can improve the model efficiency and decrease the complexity by a good factor.

References

[1] 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 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.

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

[3] Djuric N, Radosavljevic V, Cui H, Nguyen T, Chou FC, Lin TH, Singh N, Schneider J. Uncertainty-aware short-term motion prediction of traffic actors for autonomous driving. InProceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision 2020 (pp. 2095-2104).

[4] Waymo open dataset motion prediction challenge: Leaderboard. https://waymo.com/open/challenges/2021/motion-prediction, 2021. 3, 4

[5] Ettinger, Scott, Shuyang Cheng, Benjamin Caine, Chenxi Liu, Hang Zhao, Sabeek Pradhan, Yuning Chai 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, pp. 9710-9719. 2021.

[6] https://woven-planet.github.io/l5kit/

[7] Time series prediction: https://github.com/ pytorch/examples/tree/main/time_sequence_prediction

[8] “Waymo Open Dataset.” 2022. GitHub. May 2, 2022. https://github.com/waymo-research/waymo-open-dataset.

[9] “MotionCNN: A Strong Baseline for Motion Prediction in Autonomous Driving” 2022. GitHub. May 2, 2022. https://github.com/waymo-research/waymo-open-dataset.