CS230

Lyft Motion Prediction for Autonomous Vehicles

Govinda Puthalapat & Dipti Nemade
govindap@stanford.edu & diptin@stanford.edu

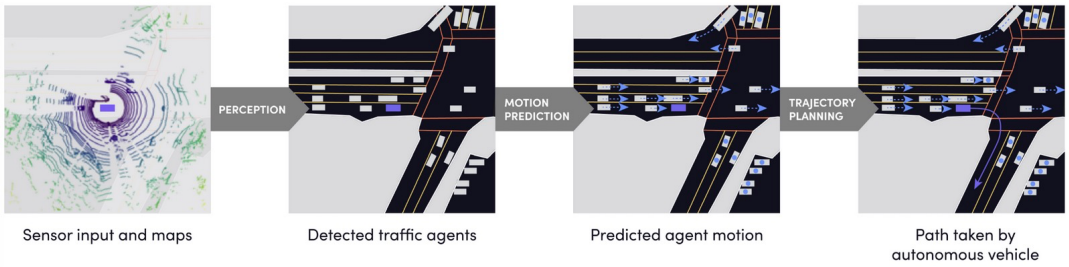
1 Problem Description

Autonomous Vehicles(AV) require a significant amount of machine learning capabilities for behavior planning. Part of the planning pipeline that includes detecting objects and classifying them is automated. Still, the planning aspects of what AV should do at any time are driven by a set of rules and remained an engineering challenge for learned trajectory planning. We will be building models for predicting the motion trajectories of the traffic agents around the AV.

2 Challenges

The challenge will be modeling the ambiguity in different agent behaviors as a point estimate and also capture uncertainties in the estimate. Analyze how behaviors of different agents influence each other in the system. Incorporating longer time range agent behaviors in the neural networks.

As shown in the picture below, the bigger task for the project is predicting agent motion given the detected traffic agents.



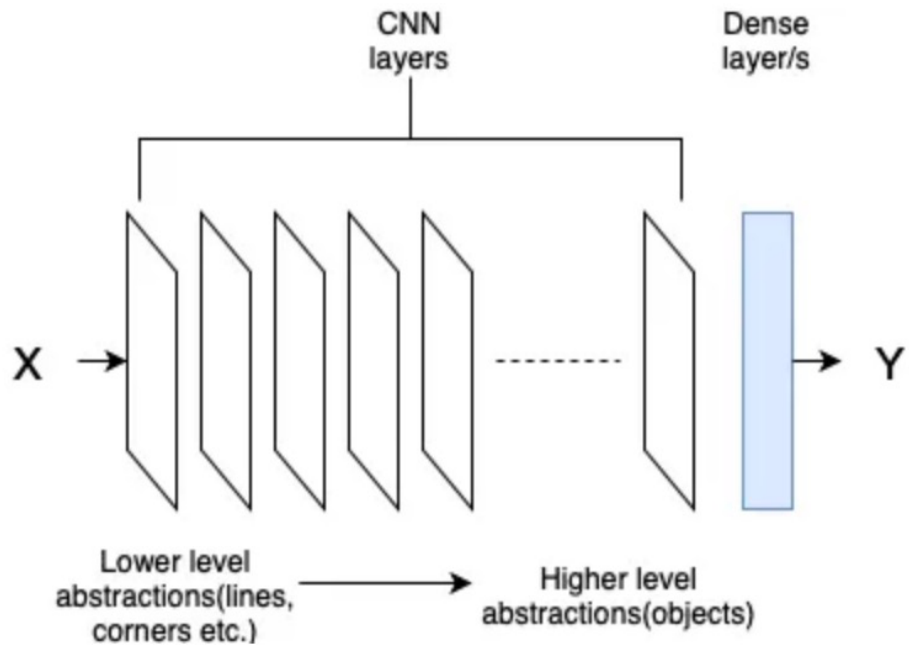
3 Dataset

The dataset includes more than 1000 hours of driving data by Lyft's AV fleet. Full set of files include: scenes: driving episodes acquired from a given vehicle. frames: snapshots in time of the pose of the vehicle. agents: a generic entity captured by the vehicle's sensors. Note that only 4 of the 17 possible agent label probabilities are present in this dataset.
agents mask: a mask that (for train and validation) masks out objects that aren't useful for training. In test, the mask masks out any test object for which predictions are NOT required.
traffic light faces: traffic light information.

4 Learning Method

Prediction task builds on top of the agent behavior to predict the trajectories in the future time period. Deep learning solution leverages the birds-eye-view(BEV) representation of the agent movement represented as a semantic map with lanes, agents, obstacles and other information observed by AV's lidars, radars and cameras.

Method1: Building model using various ResNet18/34/50 CNN architecture by stacking historical BEV of the agent history; ResNet initial weights are loaded and allowed to update the model during training. It used 224 x 224 BEV rasterized images centered around the several agents in the model. Model is evaluated on Negative Multi Log Likelihood loss by comparing the predicted future trajectories with the observed. Dense layer is added after the ResNet layers for the final logit prediction of the trajectories. (As shown in the picture below)



Method2: Convolutional LSTM architecture is used to capture spatio-temporal prediction, since agent movement captured through images has state transition. The input frame is encoded through Resnet architecture, without error propagation. The output of each frame is captured as a sequence for ConvLSTM, the LSTM layer captures hidden representations of the moving objects, larger kernel will capture faster motions, whereas smaller kernel will capture slower motions.

The key equations of ConvLSTM is shown below:

$$i_t = \sigma(W_{xi} * X_t + W_{hi} * H_{t-1} + W_{ci} \odot \mathcal{C}_{t-1} + b_i)$$

$$f_t = \sigma(W_{xf} * X_t + W_{hf} * H_{t-1} + W_{cf} \odot \mathcal{C}_{t-1} + b_f)$$

$$\mathcal{C}_t = f_t \odot \mathcal{C}_{t-1} + i_t \odot \tanh(W_{xc} * X_t + W_{hc} * \mathcal{H}_{t-1} + b_c)$$

$$o_t = \sigma(W_{xo} * X_t + W_{ho} * \mathcal{H}_{t-1} + W_{co} \odot \mathcal{C}_t + b_o)$$

$$\mathcal{H}_t = o_t \odot \tanh(C_t)$$

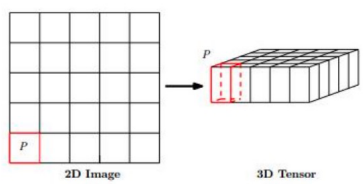


Figure 1: Transforming 2D image into 3D tensor

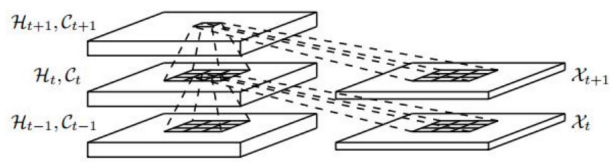


Figure 2: Inner structure of ConvLSTM

Loss:

Negative Multi Log Likelihood (NMLL) Loss function captures multi-modality of the trajectory prediction. In the Multiple-Trajectory prediction (MTP) loss, for every forward pass of the neural network we obtain three output trajectories. We then identify the mode that is closest to the ground truth by using a trajectory distance function and forcing the best matching mode as close to the true trajectory.

Hyper-Parameters:

In the experiments, the following hyperparameters are tuned.

- Different pre-trained Resnet models which contain different layers like 18, 34, 50, 152.
- The length of the past frames to include, in our experiments past 10 frames of agent/EGO history is included.
- The model is run on randomly selected data with 2000 rounds of training.
- Fully Connected (FN) layer sizes

Results:

Method1 (CNN) :

Larger models are able to capture the features well because of the large number of stacked history layers available as part of training. The model lets the gradient flow the Resnet layers of the network.

Base CNN	**Fully Connected Network Size**	**Loss**	**Score (Public Test Set)**
ResNet50	512	L2	196.6

ResNet18	512	Negative Multi Log Likelihood	46.3
ResNet50	512	Negative Multi Log Likelihood	23.6

Method2 (ConvLSTM) :

In ConvLSTM, the gradient is not propagated through the Resnet layers of the network. LSTM size and layers had large impact on the score.

Method	**Validation Set Sample Loss (NMLL) (Generalization Error)**
CNN – Resnet 50	267.8
ConvLSTM (1 layer) with Resnet 50	600.1
ConvLSTM (2 layers) With Resnet 50	480.8

5 Conclusion:

ConvLSTM based spatio-temporal model for vehicle motion prediction was able to predict multi-modal trajectories of the agents through better generalization of agent behavior and it's also to be seen if this model will score higher when the competition ends. The loss compared to non-frozen Resnet model is higher because CNN features in ConvLSTM is not optimized for the problem and allowing it to update will seem to give better results.

6 Evaluation:

uni-modal models yielding a single prediction per sample, or multi-modal ones generating multiple hypotheses (up to 3) - further described by a confidence vector will be used for evaluation. Full details for the evaluation from the Kaggle competition website shown below.

We calculate the negative log-likelihood of the ground truth data given the multi-modal predictions. Let us take a closer look at this. Assume, ground truth positions of a sample trajectory are

$$x_1, \ldots, x_T, y_1, \ldots, y_T$$

and we predict K hypotheses, represented by means

$$\bar{x}_1^k, \ldots, \bar{x}_T^k, \bar{y}_1^k, \ldots, \bar{y}_T^k$$

In addition, we predict confidences c of these K hypotheses. We assume the ground truth positions to be modeled by a mixture of multi-dimensional independent Normal distributions over time, yielding the likelihood

$$p(x_{1,\ldots,T}, y_{1,\ldots,T} | c^{1,\ldots,K}, \bar{x}_{1,\ldots,T}^{1,\ldots,K}, \bar{y}_{1,\ldots,T}^{1,\ldots,K})$$

$$= \sum_k c^k \mathcal{N}(x_{1,\ldots,T} | \bar{x}_{1,\ldots,T}^k, \Sigma = 1) \mathcal{N}(y_{1,\ldots,T} | \bar{y}_{1,\ldots,T}^k, \Sigma = 1)$$

$$= \sum_k c^k \prod_t \mathcal{N}(x_t | \bar{x}_t^k, \sigma = 1) \mathcal{N}(y_t | \bar{y}_t^k, \sigma = 1)$$

which results in the loss

$$L = -\log p(x_{1,\ldots,T}, y_{1,\ldots,T} | c^{1,\ldots,K}, \bar{x}_{1,\ldots,T}^{1,\ldots,K}, \bar{y}_{1,\ldots,T}^{1,\ldots,K})$$

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

7 Further Research:

Further experiments will be conducted by adding agent velocity, heading direction and rotation information through concatenation of the additional state vector in the FC layer. We will decode the output of FC layer and pass it on to LSTM layer after each time step to generate the trajectory at the end of LSTM decoding.

References

[1] Kaggle Lyft Competition. https://www.kaggle.com/c/lyft-motion-prediction-autonomous-vehicles/overview/description.

[2] Convolutional LSTM Network https://arxiv.org/pdf/1506.04214.pdf

[3] An LSTM-Based Autonomous Driving Model Using Waymo Open Dataset https://arxiv.org/pdf/2002.05878.pdf

[4] Deep Learning-based Vehicle Behaviour Prediction for Autonomous Driving Applications https://arxiv.org/pdf/1912.11676.pdf

[5] Fundamentals of Car Science - Pitch Roll, yaw https://carsexplained.wordpress.com/2017/02/21/fundamentals-of-car-science-pitch-and-roll/

[6] Multi-Modal Trajectory Prediction of Surrounding Vehicles with Maneuver based LSTMs https://arxiv.org/pdf/1805.05499.pdf

[7] Uncertainty-aware Short-term Motion Prediction of Traffic Actors for Autonomous Driving. https://openaccess.thecvf.com/content_WACV_2020/papers/Djuric_Uncertainty-aware_Short-term_Motion_Prediction_of_Traffic_Actors_for_Autonomous_Driving_WACV_2020_paper.pdf