Module 2: 3D Object Tracking & Motion Forecasting

CSC490H1 2022: Making Your Self-driving Car Perceive the World

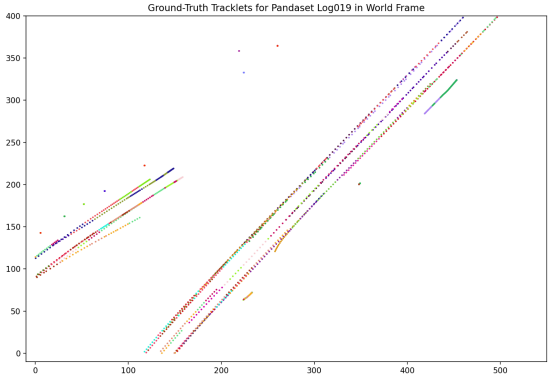
Gongyi Shi, Anny Runxuan Dai

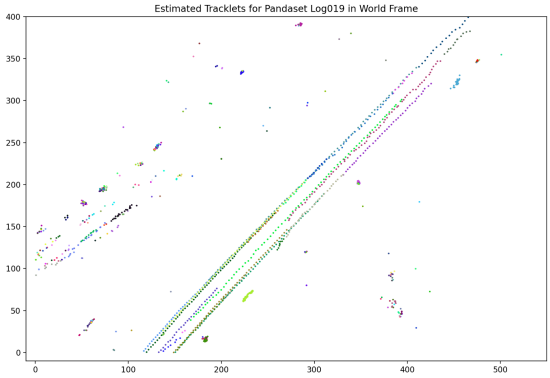
Part A: Object Tracking

Object Tracking Questions

Evaluation and Analysis

Noticeably, the evaluation outputs of the validation set are the same for both Hungarian and greedy association algorithms (See Appendix for the table). Although theoretically, the Hungarian algorithm finds the optimal matches while greedy does not, there is no difference in the association matrix the baseline implementation after filtering by a match threshold = 1.





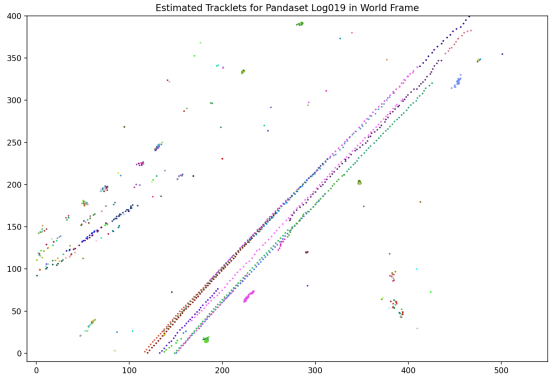


Fig. 1.1. From top to bottom: visualizations of the ground-truth, the Hungarian-associated, the greedy-associated tracklets of data with sequence ID 019.

Improved Object Tracking Report

Motivation

We choose to implement the Complete IoU loss (CIoU) proposed by Zheng et al.(2020) to track the geometry distances: the centroid and aspect ratio, and linear velocity prediction to account for the motion feature.

Problem. While the baseline tracking loss function intersection-over-union (IoU) measures only the overlap ratio, other geometric distances are also important. IoU does not consider centroid distance when the overlap area remains unchanged (Fig. 1.2). On the other hand, the same vehicle must keep a roughly unchanged shape in every short time section - i.e. in every frame. Different aspect ratios should then be measured as an aspect as well.

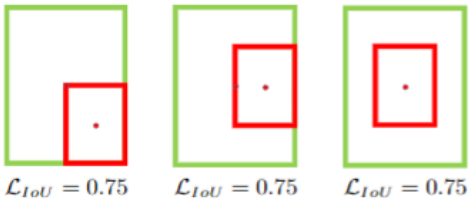


Fig. 1.2. IoU loss does not distinguish from centroid distance. Green and red denote the target box and predicted box respectively. The image was created by Zheng et al. (2020).

Furthermore, if the delta time between frames or the vehicle speed increases, for some vehicles, there can be no overlapping area in consecutive frames. In that case, any association is equivalent to the ground truth bounding box, which is not desirable.

Other approaches. A possible approach is to train a Markov chain recurrent neural network to predict the speed based on previous centroids and yaws of the vehicles, and another neural network to track the association among frames. However, such a model with IoU loss suffers from the non-moving gradient when there is no overlap area. One other possible approach could be Distance-IoU(DIoU) (Zheng, 2020), which incorporates the centroid distance. It converges faster than IoU and smoothes the gradient when the overlap area is zero but does not consider the aspect ratio. Therefore, a neural network-based tracking model with DIoU/ loss could be another approach that solves this problem.

Motivation. Our improvement on the loss function further accounts for the geometry distances (the centroid distance and aspect ratio), and the motion feature. As mentioned, the centroid distance and aspect ratio are critical aspects that the loss needs to account for, while motion prediction addresses the problem of the fast-moving object and increasing time step. While using motion forecasting is possible to aid tracking, we focus only on linear motion feature prediction, as we choose the first direction to explore.

Techniques

Intuition. When there is no overlap area, IoU treats close boxes with similar aspect ratios, and far away boxes with different aspect ratios as equally likely. However, the former is likely to be within the same tracklet intuitively, and CIoU prefers the former more than the latter. On the other hand, as IoU suffers from uninformative when no overlap area, the motion feature might prevent such a situation. As the speed of vehicles or time interval between frames increases, there might be no overlap area for consecutive bounding boxes. The motion feature predicts the box positions in the current frame and compares the IoU with the actual position, which suggests a higher chance of overlapping.

Algorithms. We explored two approaches to improving the loss value. The first is to use the CIoU loss proposed by Zheng et al.(2020). Based on IoU loss, CIoU further introduces a penalty term for the central point distance and aspect ratio for predicted box B and target box B^{gt}.

$$\mathcal{L} = 1 - IoU + \mathcal{R}(B, B^{gt}),$$

$$\mathcal{R}_{CIoU} = \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} + \alpha v, \tag{8}$$

where α is a positive trade-off parameter, and v measures the consistency of aspect ratio,

$$v = \frac{4}{\pi^2}(arctan\frac{w^{gt}}{h^{gt}} - arctan\frac{w}{h})^2. \tag{9}$$

Then the loss function can be defined as

$$\mathcal{L}_{CIoU} = 1 - IoU + \frac{\rho^2(\mathbf{b}, \mathbf{b}^{gt})}{c^2} + \alpha v. \tag{10}$$

And the trade-off parameter α is defined as

$$\alpha = \frac{v}{(1 - IoU) + v}, \tag{11}$$

by which the overlap area factor is given higher priority for regression, especially for non-overlapping cases.

where b and b^{gt} denote the central points of B and B^{gt}, $\rho(\cdot)$ is the Euclidean distance, and c is the diagonal length of the smallest enclosing box covering the two boxes. And then the

Fig. 1.2. The CIoU loss equation proposed by Zheng et al. (2020).

When two bounding boxes perfectly match, CIoU loss = 0. When two boxes are different in aspect ratio, $\alpha v \to 1$, and centroids are far away, $\rho^2(\cdot)/c \to 1$, CIoU loss $\to 3$.

The second is basic IoU with motion feature prediction. Because the time interval is short and identical among frames, we assume the velocity in the current frame to be roughly the same as in the previous frame. Thus, we let the velocity in unit time (time interval between frames) of bounding boxes at time *t-1* be the difference in positions of *t-2* and *t-1*. The predicted position at *t* is then the position at t-1 plus the difference between positions at *t-2* and *t-1*:

$$\begin{aligned} & p_t \\ &= p_{t-1} + t_{unit} \cdot v_{change} \\ &= p_{t-1} + (p_{t-1} - p_{t-2}) \\ &= 2p_{t-1} - p_{t-2} \end{aligned}$$

We then can apply IoU loss on the predicted positions at time *t* with the actual positions.

Evaluation

Steps.

1. We performed a hyperparameter tuning in the training set using both association algorithms, on the `tracker.match_th`, with step = 0.1, to filter out the associations exceeding different loss value thresholds. For CIoU, the possible range of loss value is [0, 3]. For the motion feature, the range is [0,1].
2. We selected the hyperparameter with the highest multiple object tracking accuracy (MOTA) score for both approaches. We then visualized and evaluated the result on the validation set.

We choose MOTA among other metrics because MOTA accounts for the overall error ratio by combining the rate of misses, false positives and mismatches in the association error of tracklets. Notably, the other metrics including the multiple objects tracking precision (MOTP), mostly tracked, mostly lost and partially tracked, account for the matching distance and tracking ratios of each tracklet respectively. Although they also depict some aspects of the error ratio, too, they do not explicitly measure the association error rate, which is our minimizing objective.

Results. The result metrics are identical with both greedy and the Hungarian association algorithms. We obtained the following average evaluation metrics at step 1 for different hyperparameters.

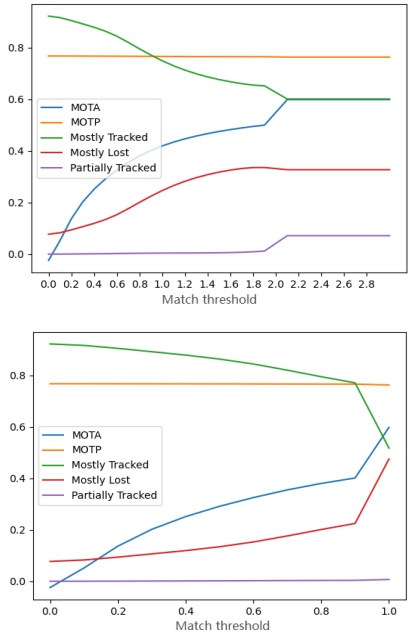


Fig. 1.3. The average evaluation metrics with different match thresholds on the training set of the CIoU (top) and the motion feature (bottom) approaches.

The metrics for CIoU remained unchanged after threshold > 2.0, so we clipped the plot. Combining with our prior knowledge of the dataset, we suggest this is due to the similar aspect ratio of bounding boxes. We then chose `tracker.match_th` = 2.1 and 1 (we added 1e-5 to prevent floating point error) respectively for the two approaches, tested on the validation dataset and obtained the final results (Fig. 1.4, see the full table in Appendix for quantitative results). There is little (<1e-3) or no difference with either association algorithm.

Approach	Mean MOTA with Hungarian
IoU	0.3242
Motion Feature	0.2660
CIoU	0.3244

Fig. 1.4.1. The mean MOTA over the validation set of the baseline IoU with default threshold, the CIoU and the motion feature with the tuned thresholds.

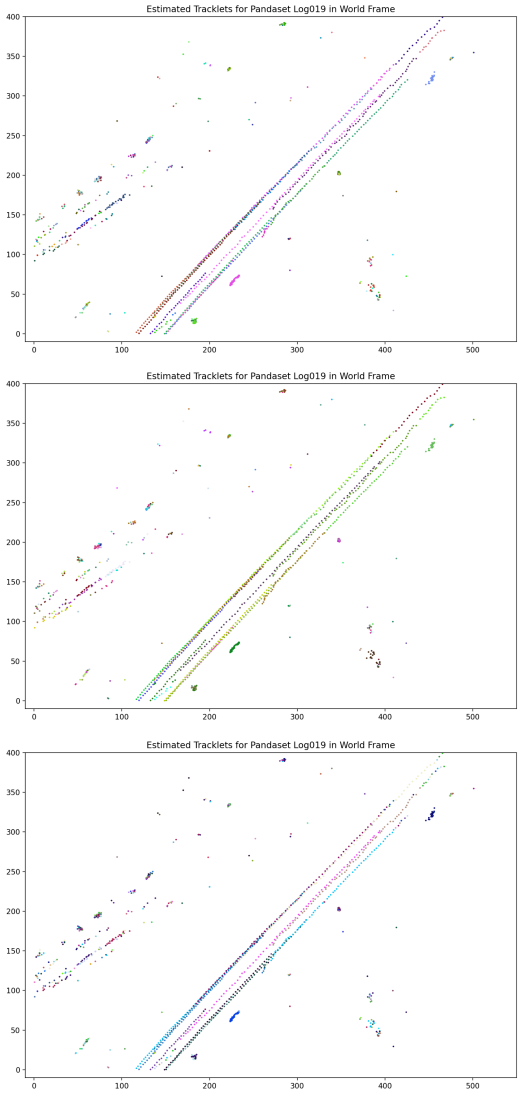


Fig. 1.4.2. The predicted tracklet visualization using the Hungarian association algorithm of the baseline IoU (top), the CIoU loss approach (middle) and the motion feature approach (bottom) of data with sequence ID 019.

The performance of CIoU overcomes the baseline implementation insignificantly, and the motion feature approach obtains worse MOTA. The results suggest that in our dataset, aspect ratio and centroid distance provides little extra information compared to ordinary IoU. The clusters in Fig 1.4.2 are identified as more tracklets by the motion feature approach. This suggests the constant velocity assumption is violated, because the time interval between frames is not short enough to obtain a roughly unchanged velocity for every consecutive frame, and/or some vehicles change direction or velocity frequently within the time interval.

Limitations

Setups. The setups are successful as planned, and both approaches show improvements in MOTA.

Limitations. There are several limitations of our current implementation as well:

1. Compared to the neural-network-based model, our implementation is too slow to be used in real-time situations because every association in consecutive frames costs more than 100 ms.
2. Our model could easily fail to associate with missing frames or missed detections. CIoU loss would be too large and get filtered out, while the motion feature becomes less reliable when using non-consecutive frames.
3. Linear velocity is not applicable to many situations, besides when the two strong assumptions are satisfied: unchanged and short time intervals, and roughly equal velocity between such time intervals. Any violations of the assumptions (E.g. missed detection, hard braking, etc.) could result in unreliable prediction.

Future works. As we suggested, we can construct a Bayesian LSTM for velocity and position predictions. We can then handle the missing data by predicting the posterior. Combining a simple MLP model, with CIoU loss, to predict the association, given the predicted velocity and position, we can achieve a faster, less constrained, and potentially better-performed model that can be applied in real-time.

Part B: Baseline Motion Forecaster

Part 1: Encoding Past Trajectory Info (Prediction Encoder)

Please refer to PredictionModel in prediction/model.py.

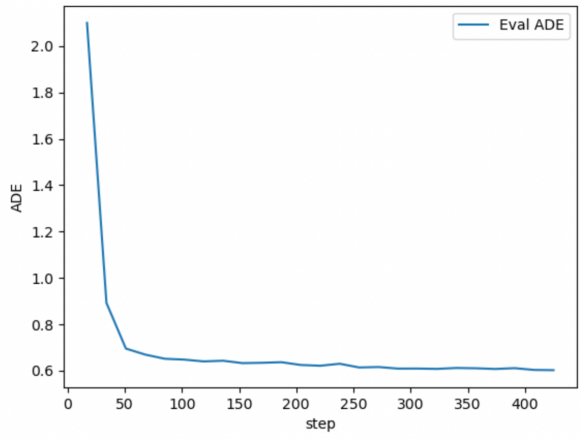
In our opinion, adding the yaw information will speed up the process of convergence. However, the final results might not change a lot since knowing the last few timesteps already hints us on the yaw of the cars.

We have trained our model with 10 input frames and an MLP with 1 layer in the encoder with and without yaw.

Here are the results:

	ADE	FDE
With yaw	0.6024	1.3366
Without yaw	0.6059	1.3413

Fig. 2.1 comparison of the ADE and FDE when training with or without yaw



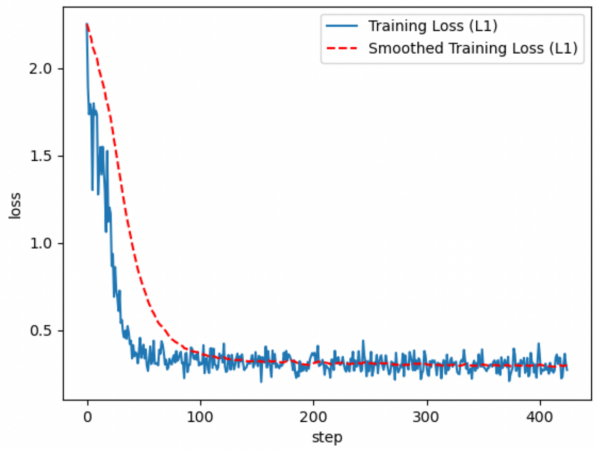
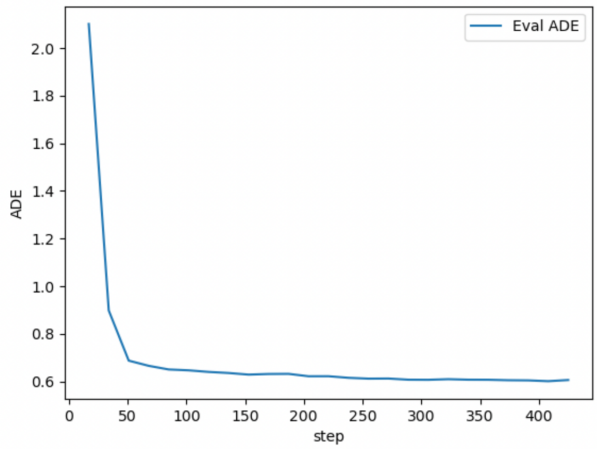


Fig. 2.2. The evaluation and training loss graph of training on MLP with 1 layer with input frames = 10 with yaw as part of the input



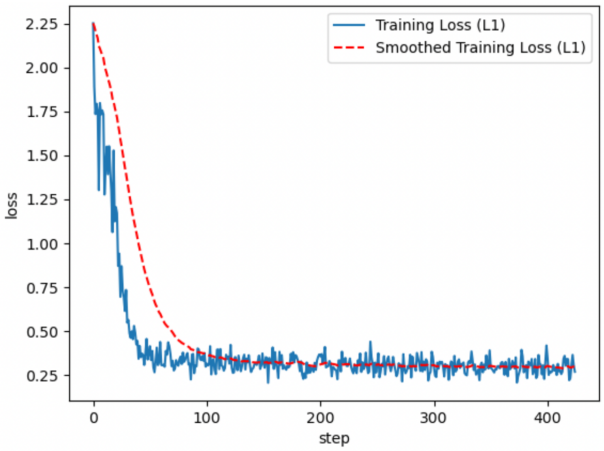


Fig. 2.3. The evaluation loss graph of training on MLP with 1 layer with input frames = 10 without yaw as part of the input

From the results, we can see that both models converge at about 150 iterations.

The resulting ADE and FDE of the model with yaw are slightly higher than the one without yaw. However, the numbers are very close. We can still conclude that adding yaw to the input is not important.

Part 2: Predicting Future Timesteps (Prediction Decoder)

Please refer to `PredictionModel` in `prediction/model.py`.

For other output parametrization, we can output the predicted velocity or yaw. The pros of doing so are that we can compare if the yaw and the velocity are in the same direction. Then, we can predict if the car is dangerous. When the yaw and the velocity are not in the same direction, the car might be slipping. Without predicting the yaws, we can only get a hint on the velocities by predicting the locations of the centroid. The cons of doing so are that we will need more computing resources to train and use the model since there are more parameters to be considered. Thus, there might be less time for the system to react to the predictions.

We can also change the output to a possible range of movement such as a gaussian heatmap. The pros of doing so are that we can predict a range of possible positions and avoid collisions with other cars with a greater chance. Also, a wider range can make the possibility of making the wrong prediction smaller since the ground-truth value has a greater chance of being covered in the range. The cons of doing so is that the gaussian heatmap might not be the best model. The possible positions of a car might not be a gaussian heatmap since sometimes the owner might be making decisions such as going straight or making a left turn.

Part 3: Loss Function

Please refer to `compute_l1_loss` in `prediction/loss_function.py`.

Part 4: Overfitting

The following is what we get after we overfit on an MLP model with one layer, num_history_timesteps equals 10, using yaw with a learning rate of 1e-2.

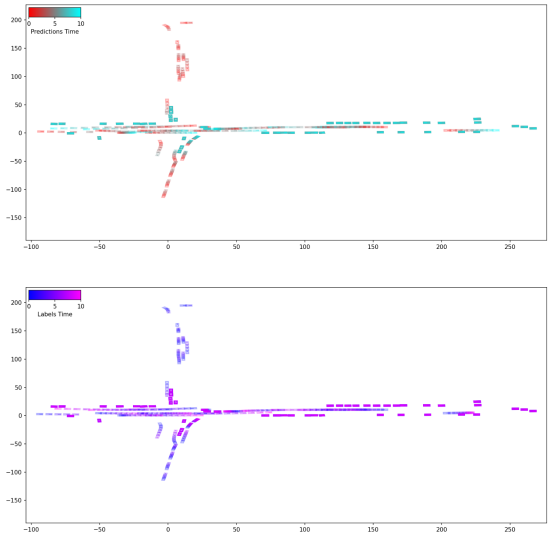


Fig. 2.4. The result of overfitting on MLP with 1 layer with input frames = 10

Part 5: Analysis

We will be analyzing our model based on the number of input frames and the architecture of the encoder.

After we trained our model, we have noticed that all models converge at about 200 iterations. Also, from the evaluation loss graphs, we did not see any sign of overfitting. Also, using the last epoch is similar to choosing the epochs before the last epoch and after the convergence. Thus, we will be comparing the parameters using the last epochs we've trained (epoch 25).

Because there is limited time to tune the parameters, we have only chosen a few reasonable numbers for different parameters.

For the number of input frames, we will be using 10, 15, and 20 to tune. For the architecture of the encoder, we will be using an encoder with one layer, two layers, or three layers.

Number of Input Frames	Architecture of the Encoder	ADE	FDE
10	1 layer	0.6024	1.3366
10	2 layers	0.5776	1.3016
10	3 layers	0.5840	1.3066
15	1 layer	0.6317	1.3998
15	2 layers	0.5948	1.3405
15	3 layers	0.5998	1.3393
20	1 layer	0.6238	1.3945
20	2 layers	0.6072	1.3629
20	3 layers	0.5908	1.3297

Fig. 2.5. The ADE and FDE values for tuning the number of input frames and the architecture of the encoder

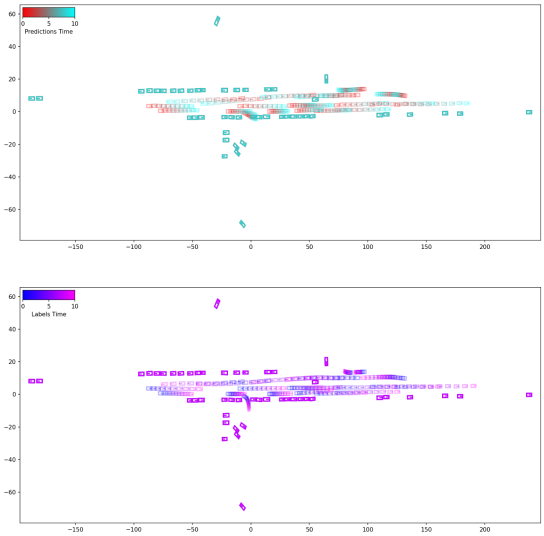


Fig. 2.7. The test result at 000 for training on 10 input frames and 1 layer of MLP

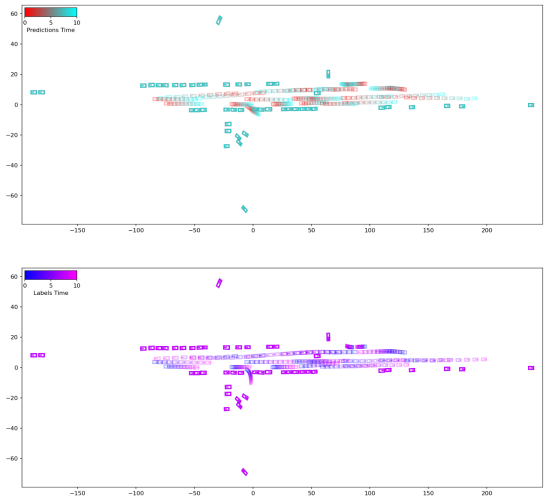


Fig. 2.8. The test result at 000 for training on 15 input frames and 2 layers.

We can see that when there are 10 input frames and 1 layer of MLP, the value of ADE and FDE are the lowest. Thus, this is the best choice for the parameters.

Also, by viewing some of the testing results, we can see that the quality of the best model (chosen by comparing the ADE and the FDE) when there are 10 input frames and 1 layer of MLP is slightly better than the one with the worst result when there are 15 input frames and 2 layers.

Improved Motion Forecaster Report

Motivation

The problem being tracked is that the prediction model before is too simple for predicting the path of a car. Notice that there are many factors that may affect the path of a car and it is very hard to train on all of them. Also, the path of a car should be following a gaussian matrix since there are also probabilities of taking other paths. Thus, our approach is assuming that the possibilities of where a car should follow a gaussian distribution.

This problem is relevant because a prediction that predicts a range instead of a point can provide more information to choose from. If the system chooses the path that has the least interaction with the predicted areas where the cars can be, it would be safer.

There are many other approaches to this problem. We can do an Interactive Predictions model where we can use a transformer model that has attention to all positions of the cars. It first predicts the possible further locations, then feeds it into another network to predict the probabilities of the locations.

We can also do a Multi-Modal Predictions model where it takes account of all possibilities and can choose the most possible paths. This way, we can handle the case where there is a 50-50 chance for the driver to choose two or more paths.

We choose this model because it should be the safest if we are predicting a range of positions of the cars. Also, a GNN model is simpler than a CNN or Transformer model used in an Interactive Predictions model or Multi-Modal Predictions model. Thus, the system should have more time to react to what was predicted.

Techniques

Our model should help because knowing more possibilities of the position of the future location of the cars can help arrange the driving positions of the car to avoid any accident. Where if we try to avoid most of the predicted locations of the car should decrease the rate of accidents.

We are going to use the following equations[2] from “SPAGNN: Spatially-Aware Graph Neural Networks for Relational Behavior Forecasting from Sensor Data”

Since our purpose is just predicting the gaussian matrix (which is also the covariance matrix) and the means of the position of the cars, the use of Von Mises distributions should not matter. Thus, we will be using a GNN model that first changes the embeddings of the input, then, we will use an MLP to predict μ_x, μ_y, σ_x, σ_y, and ρ - the means of the x- and y-axis, the variance of the x- and y-axis, and the correlation coefficient.

Inspired by GaBP, the output state $o_v^{(k)}$ at each message passing step k consists of statistics of the marginal distribution. Specifically, we assume the marginal of each waypoint and angle follow a Gaussian and Von Mises distributions respectively, *i.e.*, $p(\mathbf{x}_v^{(k)}|\Omega) = \mathcal{N}(\mathbf{x}_v^{(k)}|\boldsymbol{\mu}_v^{(k)}, \boldsymbol{\Sigma}_v^{(k)})$, $p(\theta_v^{(k)}|\Omega) = \mathcal{V}(\theta_v^{(k)}|\eta_v^{(k)}, \kappa_v^{(k)})$, where $\mathbf{x}_v^{(k)} = [x_v^{(k)}, y_v^{(k)}]^\top$, $\boldsymbol{\mu}_v^{(k)} = [\mu_{x_v}^{(k)}, \mu_{y_v}^{(k)}]^\top$,

$$\boldsymbol{\Sigma}_v^{(k)} = \begin{pmatrix} {\sigma_{x_v}^{(k)}}^2 & \rho_v^{(k)} \sigma_{x_v}^{(k)} \sigma_{y_v}^{(k)} \\ \rho_v \sigma_{x_v}^{(k)} \sigma_{y_v}^{(k)} & {\sigma_{y_v}^{(k)}}^2 \end{pmatrix}. \tag{5}$$

$$\begin{aligned} \mathcal{L}_{nll} = \sum_{i=1}^{N} \sum_{t=1}^{T} & \frac{1}{2} \log |\boldsymbol{\Sigma}_{i,t}| + \frac{1}{2} (\mathbf{x}_{i,t} - \boldsymbol{\mu}_{i,t})^{\mathrm{T}} \boldsymbol{\Sigma}_{i,t}^{-1} (\mathbf{x}_{i,t} - \boldsymbol{\mu}_{i,t}) \\ & - \kappa_{i,t} \cos(\theta_{i,t} - \eta_{i,t}) + \log \left(2\pi I_0(\kappa_{i,t})\right) \end{aligned}$$

Evaluation

In our loss function, we have clamped the value of the ρ's in the range [-1, 1] since we cannot assure that our model predicts something inside of the range. In our opinion, we think that batchsize*N is the number of data and the x centroid, y centroid, and yaw at each timestep is an embedding of the cars, so we did not add any embedding layer to the model. For all models, we will be using the following equation for GNN, where GRU is a gated recurrent unit that normalizes the results.

$$H_{t+1} = GRU(H_t, R_t)$$

$$R_t = \sum AE_k H_t$$

A is the adjacency matrix that represents which of the data are related. We will fix the number of history timesteps to 10 and the learning rate to 1e-2 to evaluate.

For generating the resulting image, we will plot the μ_x and μ_y as the centroids, σ_x and σ_y as the x and y sizes, and the ground truth yaws as the yaws. Our first approach is to use a GNN model that treats the input as H_0 and uses H_1 as the input to an MLP with one layer. Since each car does not have anything in common, we will be using the identity matrix for the adjacency matrix. The lowest ADE and FDE are 3.0580 and 5.4560 at epoch 6.

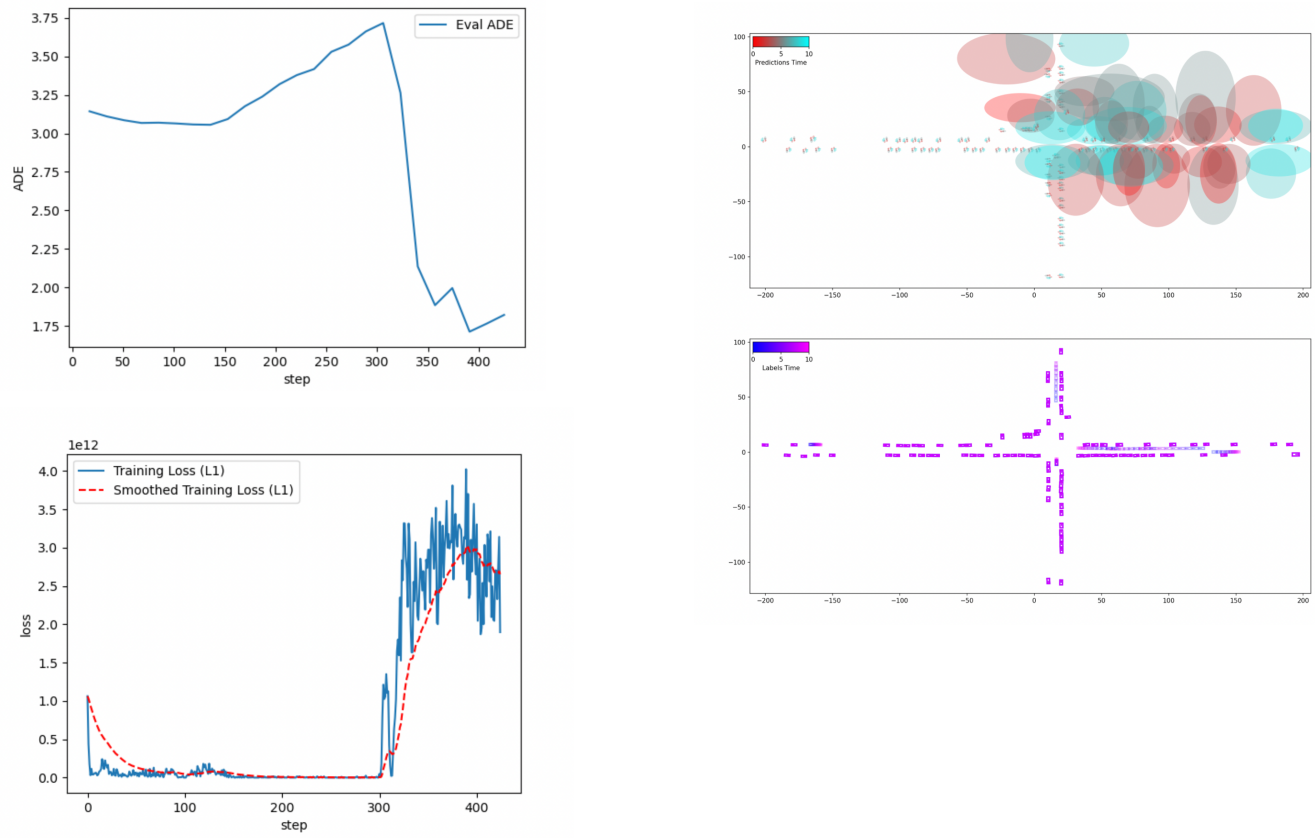


Fig. 2.2.1. The Evaluation loss, the training loss, and the prediction of our first model.

Clearly, this approach fails. However, we can still see that the mean of the cars that are staying still is quite accurate, the problem is with the ones that should be moving.

Our second approach is to use a GNN model that treats the input as H_0 and uses H_2 as the input to an MLP with one layer. We choose this model because we want to make the model more complex. Since MLP with one layer worked well in part B, we are increasing the complexity of the GNN. We will still be using the identity matrix for the adjacency matrix. The lowest ADE and FDE is 2.7870 and 6.3937 at epoch 7.

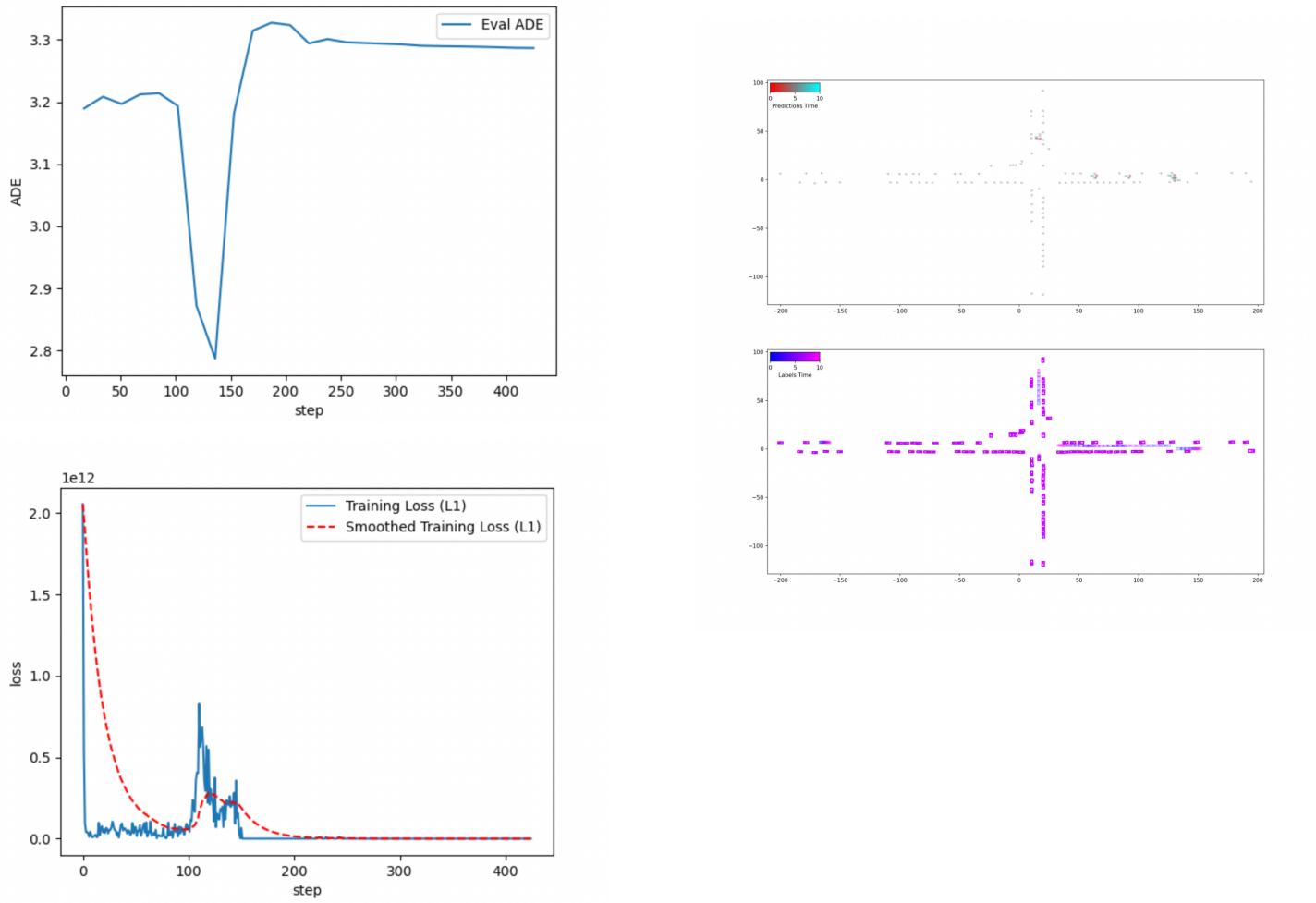


Fig. 2.2.2. The Evaluation loss, the training loss, and the prediction of our second model.

Clearly, this approach fails. However, we can still see that the mean of the cars that are staying still is quite accurate, the problem is with the ones that should be moving like model 1.

Our third approach is to use a GNN model that treats the input as H_0 and uses H_1 as the input to an MLP with two layers. We will still be using the identity matrix for the adjacency matrix. The lowest ADE and FDE are 2.6402 and 5.3933 at epoch 8.

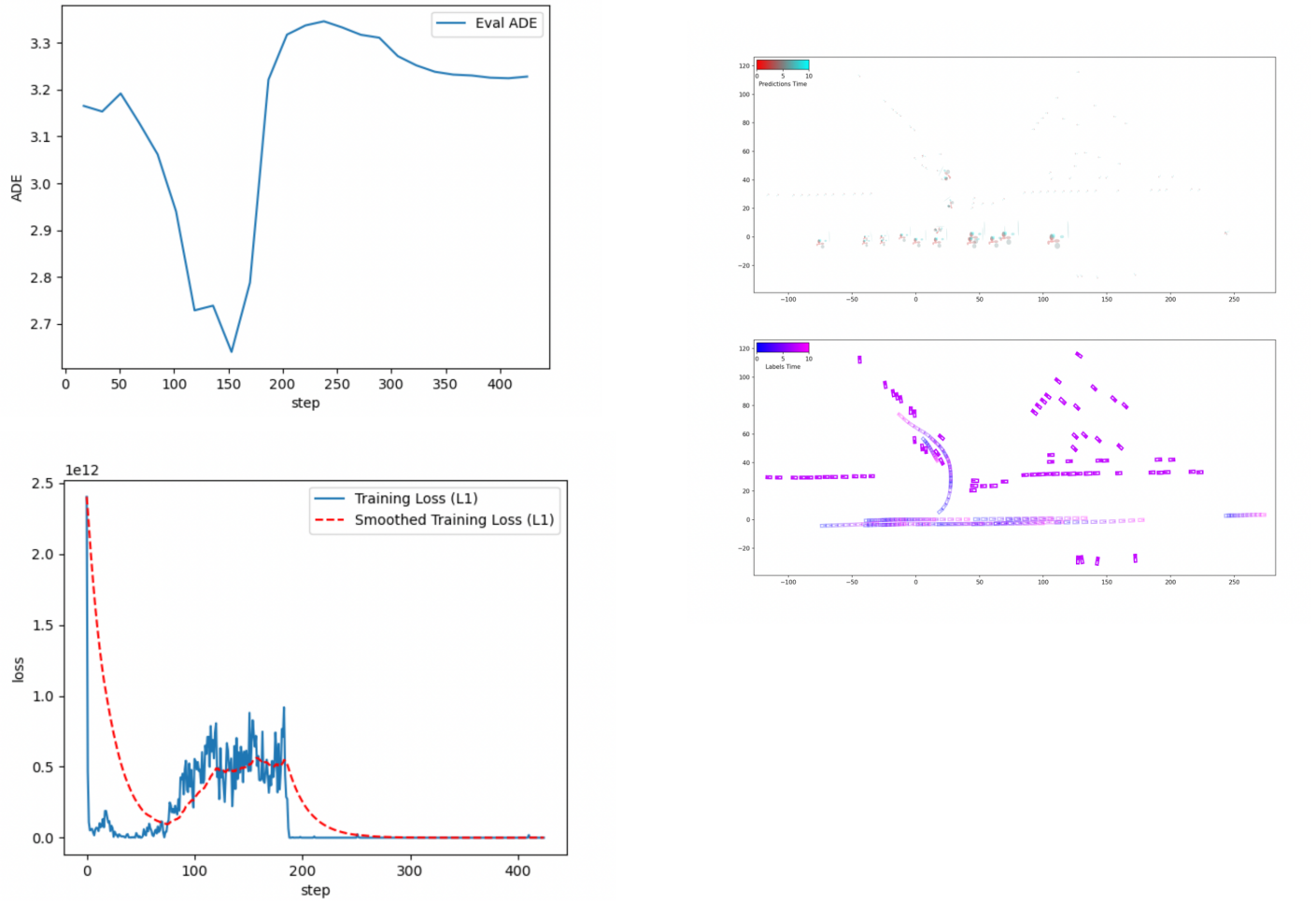


Fig. 2.2.3. The Evaluation loss, the training loss, and the prediction of our third model

We can see that the approach failed again, clearly. The problem is with the cars that should be moving like again.

Limitation

We have tried to use a GNN model that treats the input as H_0 and uses H_1 as the input to an MLP with one layer, a GNN model that treats the input as H_0 and uses H_2 as the input to an MLP with one layer. And they all failed to predict the path of moving vehicles. I think this happens because we did not add any embedding layer and the adjacency matrix is not correct.

In the future, we can improve the GNN model where we can use `torch_geometric.loader.DataLoader` to generate an adjacency matrix to increase the number of accurate tests. Also, since our model is not working, we might be doing more hyper-parameter tunings when the model has an ADE or FDE value that is smaller and the predictions make some sense.

Appendix

Part A: Object Tracking

The evaluation results over the validation set. The baseline metrics are identical for each datapooint in both the Hungarian and greedy association algorithms.

Sequence ID and Approaches	MOTA	MOTP	Mostly Tracked	Mostly Lost	Partially Tracked
Baseline IoU, match threshold = 1.0					
002	0.4757	0.5720	0.6761	0.3036	0.0202
003	0.3520	0.5692	0.7661	0.2439	0.0
004	0.2575	0.4848	0.3411	0.6124	0.0465
005	0.1572	0.6283	0.3427	0.6404	0.0169
017	0.5284	0.6533	0.6714	0.3172	0.0112
019	0.375	0.6163	0.5216	0.4741	0.0043
021	0.2453	0.6150	0.7658	0.2025	0.0316
028	0.4031	0.6923	0.6197	0.3662	0.0141
032	0.2397	0.6631	0.5671	0.4024	0.0305
033	0.2631	0.6309	0.6462	0.3255	0.0283
034	0.3256	0.6794	0.5786	0.3836	0.0377
035	0.2682	0.6093	0.5514	0.4393	0.0093
Median	0.2969	0.6223	0.5992	0.3742	0.0185
Mean	0.3242	0.6178	0.5865	0.3926	0.0208
CIoU loss, match threshold = 2.1					
Mean (Hungarian)	0.3244	0.6182	0.5504	0.2725	0.1771
Mean (Greedy)	0.3244	0.6182	0.5569	0.2701	0.1730
Motion feature, match threshold = 1.0					
Mean (Hungarian)	0.2660	0.6168	0.5291	0.2430	0.2278
Mean (Greedy)	0.2694	0.6169	0.5187	0.2424	0.2389

Reference

[1] Zheng, Z., Wang, P., Liu, W., Li, J., Ye, R., & Ren, D. (2020). Distance-IOU loss: Faster and better learning for bounding box regression. *Proceedings of the AAAI Conference on Artificial Intelligence*, *34*(07), 12993–13000. https://doi.org/10.1609/aaai.v34i07.6999

[2] Casas, S., Gulino, C., Liao, R., & Urtasun, R. (2019, October 18). SPAGNN: Spatially-Aware Graph Neural Networks for Relational Behavior Forecasting from Sensor Data. Retrieved from https://arxiv.org/pdf/1910.08233.pdf