Supplementary Materials for FEND: A Future Enhanced Distribution-Aware Contrastive Learning Framework For Long-tail Trajectory Prediction

Yuning Wang [1*], Pu Zhang [2*], Lei Bai [3], Jianru Xue [1†]
[1] Institute of Artificial Intelligence and Robotics, Xi'an Jiaotong University, China
[2] DiDi Chuxing, China
[3] Shanghai AI Laboratory, China
wangyn@stu.xjtu.edu.cn, {zhangpu94,baisanshi}@gmail.com, jrxue@mails.xjtu.edu.cn

1. Visualization Of Different Prototypes

In our framework FEND, we first do trajectory clustering to get different trajectory prototypes. To make the trajectory clustering more focused on the motion patterns of trajectories, we normalize the trajectories before clustering. Specifically, we translate the final observed trajectory points to the origin of the coordinate system. After that, we rotate the trajectories to make the velocity at the last observed time along the coordinate axis. Besides, we use the same trajectory normalization as Trajectron++ [4], which make the distribution of trajectory lengths the same across scenes. After the normalization of Trajectron++, the lengths of history trajectories will be in $[0, 2]$ and shorter than future trajectories.

In Fig. 1 we demonstrate different head trajectory prototype clusters on ETH-UCY. And in Fig. 2 we demonstrate different tail trajectory prototype clusters. We can see that the tail trajectory prototypes are with various motion patterns and are hard to predict.

Figure 3 shows the distribution of mean FDEs of the clusters for univ dataset. We can see that the samples are grouped into clusters by their hardness, generating some easy clusters with mean FDEs like 0.1 and some hard clusters with mean FDEs like 0.9 and 2.0.

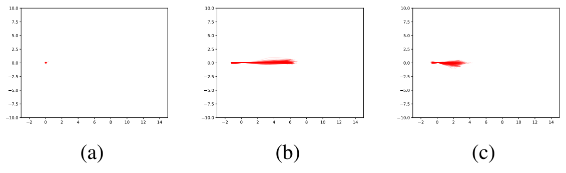

Figure 1. Trajectories in different head prototype clusters.

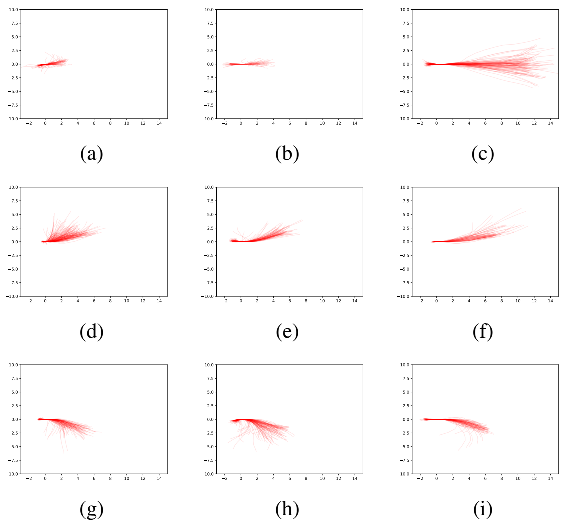

Figure 2. Trajectories in different tail prototype clusters.

2. Preliminary Implement Results On Another Baseline.

In Tab. 1 we show a preliminary implementation of our method on another backbone: Social-STGCNN [3]. The results in Tab. 1 are calculated by our implementation based on the public code of [3], and the results are calculated by taking an average of five individual runs considering the sampling randomness. We can see from Tab. 1 that our method can significantly improve the performances on all tailed samples, while the averaged performances are only very slightly affected. Those results show the generalization ability of our method. Doing contrastive learning will probably break the neighborhood correlations slightly, and we will further work on this problem in the future.

	Top 1%	Top 2%	Top 3%	Top 4%	Top 5%	All
Social-STGCNN [3]	1.90/3.97	1.61/3.43	1.45/3.14	1.33/2.89	1.25/2.72	**0.45/0.76**
Social-STGCNN+FEND	**1.73/3.69**	**1.49/3.23**	**1.35/2.96**	**1.27/2.76**	**1.20/2.59**	0.46/0.78

Table 1. Preliminary implement results of our FEND module on another baseline: Social-STGCNN. Results are in format of (minADE/minFDE) in meters. Bold numbers are the best results of each column.

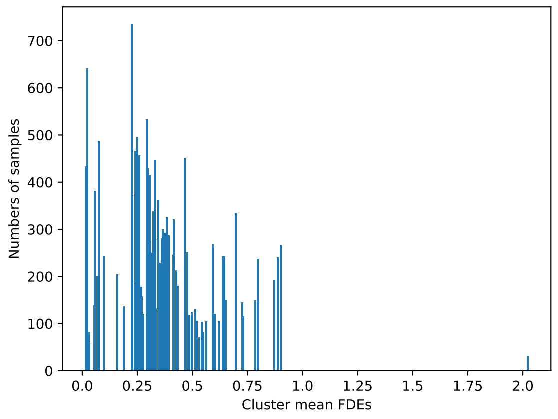

Figure 3. Distribution of the mean FDEs (from Traj++ EWTA) of different clusters. When two clusters have very similar FDEs, their bars will overlap.

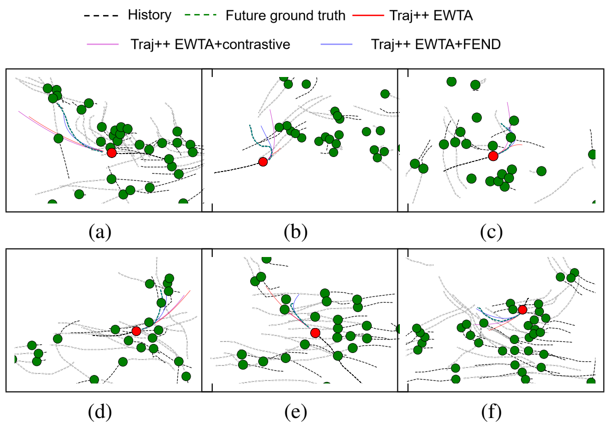

Figure 4. Qualitative visualizations of our model versus Traj++ EWTA and Traj++ EWTA + contrastive .

3. More Visualization Results

In Fig. 4 we demonstrate more qualitative comparisons of our method in ETH-UCY dataset with Traj++ EWTA [2] and Traj++ EWTA + contrastive [2]. We can see that in all those challenging scenarios our method outperforms those two methods.

θ	Top 1%	Top 3%	Top 5%	All
0	**0.84/2.12**	0.62/1.48	0.53/1.21	0.17/0.32
0.2	0.84/2.13	**0.61/1.46**	**0.52/1.19**	**0.17/0.32**
0.5	0.91/2.32	0.64/1.58	0.54/1.27	0.17/0.32

Table 2. Study on the parameter sensitivity of the threshold θ .Results are in the format of (minADE/minFDE) in meters.

4. More Parameter Sensitivity Study

Table 2 shows the parameter study of the head sample filter threshold θ to apply PCL auxiliary loss. We can see that adding $\theta = 0.2$ outperforms the model without a threshold, but filtering out too much samples will harm the performance of the tail samples, as $\theta = 0.5$ shows.

5. Performances On Kalman Filter Scores

Our method can also perform decently on another hardness score: Kalman Filter prediction errors. Table 3 contains the results for the Kalman scores versus [2], which is the state-of-the-art long-tail trajectory prediction method. T++E means Traj++ EWTA. Note that the Top 1% hard samples selected by Kalman prediction scores have a lower mean FDE than the Top $2-5\%$ samples, indicating that the Kalman errors can not reveal the sample hardness with respect to the neural network predictor.

6. More Ablation

In Tab. 4 we show more ablation results of our method with regard to the hypernetwork. We can see that using both hypernetwork and PCL can help with improving performances on tail samples.

7. Baseline Implement Details

We do LDAM [1] experiments using the suggested setting in [2]. The dataset is divided into 13 classes according to Kalman errors. Then an MLP classification head is added after the trajectory embeddings to predict the class divisions as an auxiliary task. The main hyperparameters are s in LDAM loss and w for task loss reweighing. We tried different values of the main hyperparameters, and finally use the $s = 1$, $w = 1$ setting, the same as the suggestion in [2].

	Top 1%	Top 2%	Top 3%	Top 4%	Top 5%	All
T++E contrastive [2]	0.38/**0.71**	0.48/1.03	0.46/1.03	0.45/1.00	0.42/0.90	0.17/0.32
T++E FEND	**0.38**/0.74	**0.43/0.92**	**0.40/0.85**	**0.39/0.82**	**0.37/0.76**	**0.17/0.32**

Table 3. Prediction errors on hard tail samples selected by Kalman prediction errors on ETH-UCY. Results are in the format of (minADE/minFDE) in meters. Bold numbers mean the best results of each column.

	Top 1%	Top 2%	Top 3%	Top 4%	Top 5%	All
Traj++ EWTA [2]	0.98/2.54	0.79/2.07	0.71/1.81	0.65/1.63	0.60/1.50	**0.17/0.32**
Traj++ EWTA + hypernetwork	0.97/2.46	0.78/2.00	0.69/1.72	0.62/1.54	0.57/1.40	0.17/0.33
Traj++ EWTA + hypernetwork +PCL	0.90/2.28	0.72/1.87	0.65/1.61	0.58/1.43	0.54/1.30	**0.17/0.32**
Traj++ EWTA + FEND	**0.84/2.13**	**0.68/1.68**	**0.61/1.46**	**0.56/1.30**	**0.52/1.19**	**0.17/0.32**

Table 4. Ablation studies of using the hypernetwork. Results are in format of (minADE/minFDE) in meters. Bold numbers are the best results of each column.

	Top 1%	Top 2%	Top 3%	All
T++ [4]	8.23	6.37	5.52	-1.12
T++ FEND	**7.97**	**6.23**	**5.31**	**-1.12**

Table 5. Full FDE NLL results on Trajectron++ on ETH-UCY. Bold numbers are the best results of each column.

	Top 1%	Top 2%	Top 3%	All
T [2]	1.85/4.63	1.44/3.69	1.23/3.15	0.19/0.34
T+contrastive	1.45/3.27	1.11/2.54	0.95/2.17	0.19/0.32
T w/o RS+FEND	**1.24/2.48**	**0.94/1.88**	**0.81/1.63**	**0.18/0.27**

Table 6. Results of 6 timesteps prediction on Nuscenes. Results are in format of (minADE/minFDE). Bold numbers are the best results of each column.

8. Performances On Another Metric

Except the minADE/FDE, our method is a plug-in module and can be evaluated with any better baselines and metrics. We plug our module into a probabilistic method : Trajectron++ (T++) [4], and evaluate using full KDE NLL. The result is in Table 5.

9. Performances On Different Prediction Time Periods on Nuscenes

We found out that the Trajectron++ EWTA model is trained with 6 as the prediction horizon but tested with 8 as the prediction horizon. We also provide the results for testing with 6 future timesteps in Tab. 6. T means Trajectron++ EWTA.

References

[1] Kaidi Cao, Colin Wei, Adrien Gaidon, Nikos Arechiga, and Tengyu Ma. Learning imbalanced datasets with label-distribution-aware margin loss. *Advances in neural information processing systems*, 32, 2019. 2

[2] Osama Makansi, Özgün Çiçek, Yassine Marrakchi, and Thomas Brox. On exposing the challenging long tail in future prediction of traffic actors. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13147–13157, 2021. 2, 3

[3] Abduallah Mohamed, Kun Qian, Mohamed Elhoseiny, and Christian Claudel. Social-stgcnn: A social spatio-temporal graph convolutional neural network for human trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14424–14432, 2020. 1, 2

[4] Tim Salzmann, Boris Ivanovic, Punarjay Chakravarty, and Marco Pavone. Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In *European Conference on Computer Vision*, pages 683–700. Springer, 2020. 1, 3