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

Abstract

Predicting the future trajectories of the traffic agents is a gordian technique in autonomous driving. However, trajectory prediction suffers from data imbalance in the prevalent datasets, and the tailed data is often more complicated and safety-critical. In this paper, we focus on dealing with the long-tail phenomenon in trajectory prediction. Previous methods dealing with long-tail data did not take into account the variety of motion patterns in the tailed data. In this paper, we put forward a future enhanced contrastive learning framework to recognize tail trajectory patterns and form a feature space with separate pattern clusters. Furthermore, a distribution aware hyper predictor is brought up to better utilize the shaped feature space. Our method is a model-agnostic framework and can be plugged into many well-known baselines. Experimental results show that our framework outperforms the state-of-the-art long-tail prediction method on tailed samples by 9.5% on ADE and 8.5% on FDE, while maintaining or slightly improving the averaged performance. Our method also surpasses many long-tail techniques on trajectory prediction task.

1. Introduction

Trajectory prediction is of great importance in autonomous driving scenarios [27]. It aims to predict a series of future positions for the agents on the road given the observed past tracks. There have been many recent methods in trajectory prediction, both unimodal [1, 48] and multimodal [10, 37, 38, 49].

Despite the high accuracy those prediction methods have achieved, most of them treat the samples in the datasets equally in both training and evaluation phases. But there is a long-tailed phenomenon in prevalent datasets [28]. For example, in real traffic scenes, most of the trajectories follow certain simple kinematic rules, while deviating and collision-avoiding circumstances are scarce. Therefore, the frequent cases are often simple and easy to predict, while the tail cases are often complicated with many motion patterns and suffer from large prediction errors, which makes them more safety-critical, as shown in Fig. 1 for the univ dataset. Despite of its significance, the long-tail prediction problem have been rarely discussed in literature.

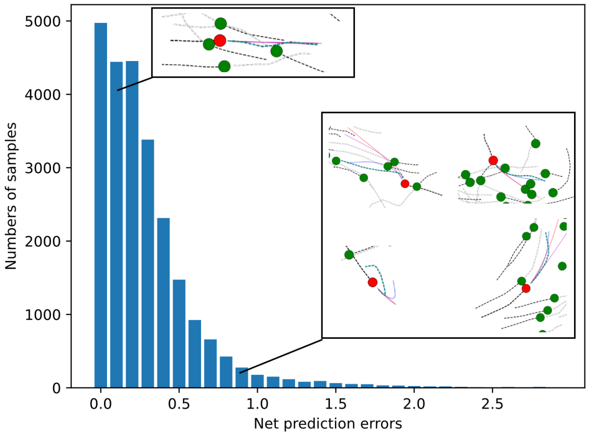


Figure 1. The long-tailed final displacement errors of the state-of-the-art prediction network: Trajectron++ EWTA [28] on ETH-UCY. The long-tail part of the dataset contains various complicated motion patterns, and predicting them is challenging.

It has been pointed out that the feature encoders largely suffer from long-tail data. In the training process, the head samples are encountered more often and dominate the latent space, while the tailed samples will be modeled insufficiently, as discussed in [24, 28, 39]. Feature embeddings of the tailed data can even be mixed up with the ones of the head data as discussed in [28], therefore the performances of the tailed samples could be harmed.

[*] Equal contributions.
[†] Corresponding author.

In this paper, we pick up the general idea of using contrastive learning to enhance the model ability on long-tailed data. A new framework is developed called FEND: Future ENhanced Distribution-aware contrastive trajectory prediction, which is a pattern-based contrastive feature learning framework enhanced by future trajectory information. An offline trajectory clustering process and prototypical contrastive learning are introduced for recognizing and separating different trajectory patterns to boost the tail samples modeling. To deal with the afore mentioned problem, the features of trajectories within the same pattern cluster are pulled together, while the features from different pattern clusters will be pushed apart. Moreover, a more flexible network structure of the decoder is introduced to exploit the shaped feature embedding space with different pattern clusters. Our contribution can be summarized as follows:

- We propose a future enhanced contrastive feature learning framework for long-tailed trajectory prediction, which can better distinguish tail patterns from head patterns, and the different patterns are represented by different cluster prototypes to enhance the modeling of the tailed data.
- We propose a distribution-aware hyper predictor, aiming at providing separated decoder parameters for trajectory inputs with different patterns.
- Experimental results show that our proposed framework can outperform start-of-the-art methods.

2. Related Work

2.1. Trajectory Prediction

Deep learning has become a mainstream trajectory prediction method because of its powerful representational ability. Some studies [1, 32, 43, 46, 48] focus on better modeling subtle relationship such as social interactions to make their prediction more precise, and some works [29, 33, 35, 36, 50] aim to produce more diverse trajectory proposals. Strong baselines [30, 38, 40, 47] have been brought up. Although the trajectory prediction methods become increasingly accurate, the long-tail issue in the task of trajectory prediction has been rarely discussed.

Trajectory prediction approaches based on clustering. Existing methods [5, 42, 44] have used trajectory clustering for trajectory prediction. MultiPath [5] performs Kmeans with the square distances between the trajectories to get anchor trajectory sets. PCCS-Net [42] decouples multimodal trajectory prediction into three steps: feature clustering, cluster selecting, and synthesizing. Memo-Net [44] clusters trajectories in the original coordinates and uses an attention network for better cluster selecting. All existing methods that use trajectory clustering are aiming at selecting future modalities for trajectory decoders and producing more diverse trajectories, which is different from our goal to distinguish tail patterns from head patterns and optimize the feature embedding space.

Trajectory prediction approaches based on contrastive learning. Contrastive learning [34] is a self-supervised method to improve the representation ability of the network given the similarities between sample pairs, and has many variants [4, 8, 19, 20] with different ways of selecting positive and negative samples and calculating contrastive loss. Prototypical Contrastive Learning (PCL) [23] is a variant of contrastive learning that can preserve local smoothness therefore induce semantically hierarchical clustered feature space [23]. Contrastive learning has also been incorporated into trajectory prediction. DisDis [7] uses contrastive learning in a CVAE framework to discriminate the latent variable distributions and make the predictions more diverse. ABC+ [12] uses action labels from their datasets and contrasts according to them. Social-NCE [26] uses contrastive learning to make the predictions away from their simulated collision cases. None of those above-mentioned methods have discussed long-tail prediction. The most relevant work is from Makansi *et al.* [28], which also tries to solve the long-tail prediction problem with contrastive learning and uses Kalman prediction errors to select positive and negative samples. Makansi *et al.* [28] push all the tailed samples together in their method. In this work, we not only separate the tails from the heads as the study [28] did, but also recognize the patterns of the tailed samples due to the fact that the tailed samples can be tailed in different ways, *e.g.* turning or accelerating, as shown in Fig. 1 and Fig. 3, which further improves the model capabilities.

2.2. Long-tailed Learning

Long-tailed learning aims to improve the performance on tailed samples when faced with unbalanced data. Most of them focus on classification tasks. Typical methods do data resampling [6, 13, 41] or loss reweighting [9, 15, 25] to improve the capability of the network on tailed samples. Recent advances [3, 31] seek for a theoretical balance of head-tail performance by means of adjusting the classification boundaries, whereas these methods cannot be directly used in regression tasks. Very recently Yang *et al.* [45] have investigated imbalanced regression tasks and propose a feature distribution smoothing and label distribution smoothing method. But the methodology in [45] needs labels with structured relationships, which is incongruent with the trajectory data. In our methods, we find out structured relationships between trajectories by forming pattern clusters, and optimize feature space according to item. Besides, we use Hypernetwork [11] as the trajectory decoder to deal with tail samples utilizing its distribution-aware modeling ability, which has not been discussed in long-tail regression to our best knowledge.

Codes available at https://github.com/ynw2021/FEND.

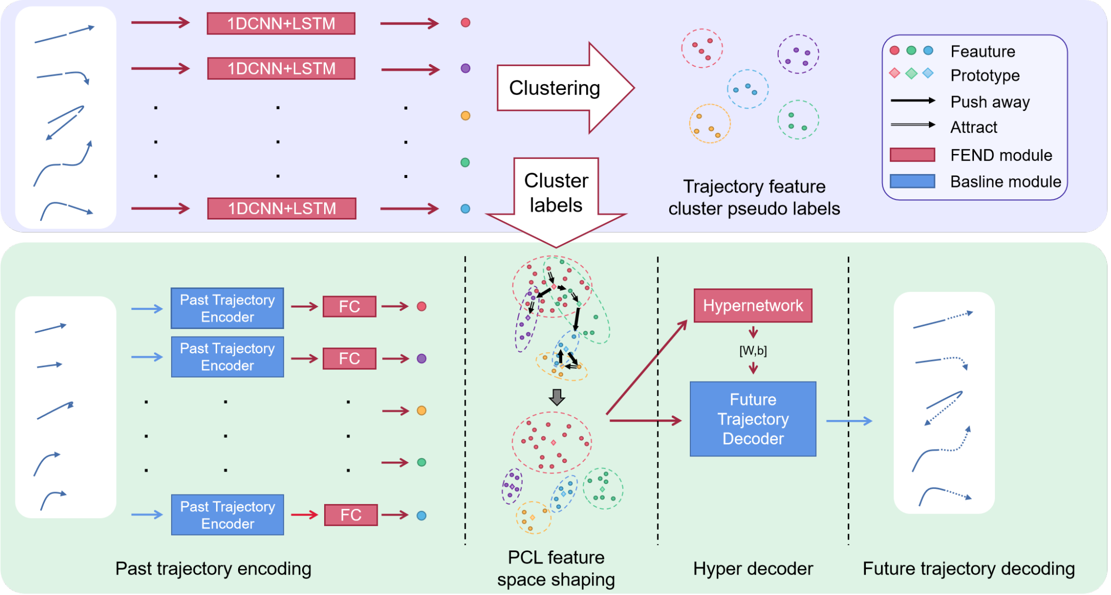


Figure 2. Illustration of our overall future enhanced distribution-aware contrastive learning framework. Top: Offline Kmeans clustering for pseudo cluster labels. Bottom: The baseline prediction network with FEND plugged in for prediction. The FEND module contains a PCL optimization procedure and a hyper decoder.

3. Method

Problem formulation. Trajectory prediction is a kind of sequential prediction problems. Given a series of past observed coordinates $\{(x_t^n, y_t^n)\}_{n=1}^N$ for N agents over time $t = -T_{obs} + 1, -T_{obs} + 2, ..., 0$, our objective is to predict the future locations $(\hat{x}_t, \hat{y}_t)$ of the agent of interest in a constant period $t = 1, 2, ..., T_{pred}$.

As discussed above, the trajectory data suffers from the long-tail phenomenon. To address this issue, we come up with a new long-tail trajectory prediction framework FEND, which contains *a future enhanced contrastive learning method* for helping shape better feature embedding for trajectory encoders, and *a more flexible distribution-aware hyper predictor* for impairing the influence from the head samples to the tail samples.

Overview. The overall framework of FEND is discribed in Fig. 2. Both the history and the future trajectories are firstly processed by a trajectory feature extractor, and the extracted features are clustered by Kmeans to form different pattern clusters. After clustering, the tail trajectory patterns and the head trajectory patterns are separated spontaneously using both history and future information. According to the pseudo cluster labels generated by Kmeans, PCL is performed on the history encoding features of the baseline prediction network. By performing PCL, the feature space of trajectory encoders is separately clustered. Then a hyper decoder is constructed which generates separate decoder weights for different trajectory inputs, therefore trajectories in the head clusters and the tail clusters are predicted differently.

3.1. Future Enhanced Contrastive Learning

3.1.1 Future Enhanced Trajectory Clustering

For trajectory pattern clustering, the start points and initial directions of trajectories should be normalized to make the feature extractor more focused on the different patterns of trajectories. But the data-preprocessing ways of present trajectory prediction methods are various. Therefore, to make our framework can be more generally applied, we use an offline cluster module to do normalization and perform trajectory clustering. Also, many trajectory prediction baselines do not have a future trajectory encoder, and their encoded past trajectory features are high-dimensional, so online clustering will be time-consuming.

We simply use a 1D convolution network (CNN) attached by an LSTM as the trajectory feature extractor for trajectory encoding and reconstruction, which is supervised by the reconstruction loss as autoencoders. The feature at the bottleneck of the network is used to perform hierarchical Kmeans. Kmeans [14] is a computation-efficient classical clustering method and can be replaced with any other clustering algorithm. We perform Kmeans with multiple level of clusters for achieving hierarchy, as the original PCL does. In the training phase of feature extractors, we also use the original PCL [23] with EM steps as an auxiliary loss to get a hierarchically clustered feature space in a self-supervised manner, which will be discussed in Sec. 3.1.2.

3.1.2 Prototypical Contrastive Learning

In our methods, we have already got the cluster labels after the trajectory clustering step. Therefore we use the cluster assignments as pseudo labels for computing prototypes and densities. The original PCL [23] is an self-supervised methods with EM steps, therefore it needs to perform clustering before every training epoch. Our methods use the pseudo labels to reduce the clustering steps therefore require less computation source compared to the original PCL. Given pseudo cluster labels, PCL can pull the features of instances belonging to the same cluster together and push the features of instances in different clusters apart, as what vanilla contrastive learning does to the positive and negative samples.

Implementing PCL loss. We do PCL on the features at the bottleneck of the encoder-decoder trajectory prediction network: after the encoder. Similar to Makansi *et al.* [28], we add a fully-connected (FC) layer after the encoder and add the PCL loss to its output features. The features before the FC layer will be given to the trajectory decoder. We perform a multi-level clustering with M hierarchies when calculating PCL loss. The PCL loss is as follows:

$$\mathcal{L}_{ProtoNCE} = \mathcal{L}_{ins} + \mathcal{L}_{proto}, \tag{1}$$

where the first term is an instance-wise contrastive term and the second-term is an instance-prototype contrastive term.

Instance-wise term. The first term in Eq. (1) is an instance-wise contrastive term considering the pseudo cluster labels, which can be written as follows:

$$\mathcal{L}_{ins} = -\sum_{i=1}^{r} \frac{1}{N_{\mathbf{po}_i}} \sum_{i_+=1}^{N_{\mathbf{po}_i}} \log \frac{\exp\left(v_i \cdot v_{i_+}/\tau\right)}{\sum_{j=1}^{r} \exp\left(v_i \cdot v_j/\tau\right)}. \tag{2}$$

The instance-wise term can help the instances gather together faster and the algorithm converge faster. v_i and v_{i_+} are feature embeddings of trajectory instance i and positive sample $i+$ after the encoder respectively, $i_+ \neq i$. $N_{\mathbf{po}_i}$ is the number of positive samples to i in a batch. τ is the contrastive temperature of the instance-wise contrastive term. In Eq. (2), the positive samples $i+$ are the instances from the same cluster with the instance i, and the rest instances in the batch, *i.e.* belonging to other clusters, are regarded as negative samples. j means an arbitrary sample in the current batch data. r denotes the batch size.

Instance-prototype term. The second term in Eq. (1) is an instance-prototype contrastive term, which can be written as follows:

$$\mathcal{L}_{proto} = -\frac{1}{M} \sum_{i=1}^{r} \sum_{m=1}^{M} \log \frac{\exp\left(v_i \cdot c_s^m/\phi_s^m\right)}{\sum_{j=1}^{N_m} \exp\left(v_i \cdot c_j^m/\phi_j^m\right)}. \tag{3}$$

The prototypes help preserving local smoothness and the formation of clusters with different patterns. In Eq. (3), M is the number of Kmeans clustering hierarchies, c_s^m means the prototype of the cluster to which i belongs, and c_j^m means the prototype of an arbitrary cluster j. The prototype is calculated by taking an average of all the features in a cluster. N_m denotes the number of clusters for hierarchy m. ϕ_j^m denotes the density of a cluster j, which is calculated as below:

$$\phi = \frac{\sum_{z=1}^{Z} \|v'_z - c\|_2}{Z \log(Z+\alpha)}, \tag{4}$$

where Z is the number of instances in the cluster, and α is a smoothing factor to ensure that small clusters do not have an overly large ϕ. We set $\alpha = 10$ same as [23]. v'_z is the momentum updated feature for instance z to ensure stability.

3.2. Distribution-Aware Hyper Predictor

Distribution-aware hypernetwork. Intuitively, the head clusters and the tail clusters should be assigned different decoders to impair their influence on each other. But there is an insufficient amount of data for the tail samples, and separately training decoders for them will cause badly overfitting. Therefore, we want to transfer common knowledge across the whole dataset, while keep the modeling flexibility of separate decoders. HyperNetworks [11] is an approach of using a small network, which is known as a hypernetwork, to generate the weights of the main network, and it naturally suits our demands. The hypernetwork contains the knowledge of all samples, which prevents overfitting. Also, there are separate decoder parameters for head and tail clusters, which make the decoder *aware of the distribution of the clustered feature space*. So the hyper decoder can predict the tailed clusters differently.

LSTM trajectory decoder. As an example of a hyper predictor, we employ an LSTM as the trajectory decoder, which is commonly used in recent studies [28, 38, 49].The original formulation of an LSTM is as follows:

$$\begin{aligned}
i_t &= W_h^i h_{t-1} + W_x^i x_t + b^i, \\
g_t &= W_h^g h_{t-1} + W_x^g x_t + b^g, \\
f_t &= W_h^f h_{t-1} + W_x^f x_t + b^f, \\
o_t &= W_h^o h_{t-1} + W_x^o x_t + b^o, \\
m_t &= \sigma\left(f_t\right) \odot m_{t-1} + \sigma\left(i_t\right) \odot \psi\left(g_t\right), \\
h_t &= \sigma\left(o_t\right) \odot \psi\left(m_t\right),
\end{aligned} \tag{5}$$

where i, g, f, o are the input gate, update gate, forget gate, and output gate respectively. $W_h \in \mathbb{R}^{N_h \times N_h}, W_x \in \mathbb{R}^{N_h \times N_x}, b \in \mathbb{R}^{N_h}$, N_h and N_x are the dimensions of input and hidden states. h_t, m_t are the hidden state and the cell state. σ is the *sigmoid* operator, and ψ is the *tanh* operator. The initial x and h are produced by the feature embedding

v of the observed trajectory:

$$\begin{aligned} x_1 &= W_x^v v + b_x^v, \\ h_0 &= W_h^v v + b_h^v, \end{aligned} \tag{6}$$

where $W_h^v \in \mathbb{R}^{N_h \times N_v}, W_x^v \in \mathbb{R}^{N_x \times N_v}, b_h^v \in \mathbb{R}^{N_h}, b_x^v \in \mathbb{R}^{N_x}$.

HyperLSTM. In our implement, the formulation of an LSTM with a small hypernetwork is as follows:

$$y_t = LN\left(d_h^y \odot W_h^y h_{t-1} + d_x^y \odot W_x^y x_t + b^y\left(z_b^y\right)\right), \tag{7}$$

where

$$\begin{aligned} d_h^y\left(z_h\right) &= W_{hz}^y z_h, \\ d_x^y\left(z_x\right) &= W_{xz}^y z_x, \\ b^y\left(z_b^y\right) &= W_{bz}^y z_b^y + b_0^y. \end{aligned} \tag{8}$$

In Eq. (7), y means one of $\{i, g, f, o\}$ four gates in the original LSTM formulation Eq. (5) for brevity. $\odot$ denotes the element-wise product operation, $LN()$ denotes the layer normalization, ds and b are the weights and bias adjusting vectors from the hypernetwork to change the weights and bias in the original LSTM. ds and b are generated by the output zs of the hypernetwork as in Eq. (8), where Ws and b_0^y are the weights and bias of the linear fully-connected layers. z can be written as follows for instance i with input feature v_i:

$$z_i = f_H\left(v_i\right), \tag{9}$$

where the f_H means the hypernetwork mapping function, which should be a shallow network to reduce computation and prevent overfitting.

3.3. Loss Reweighting

Our final network loss is as follows:

$$\mathcal{L} = \mathcal{L}_{pred} + \lambda \mathcal{L}_{ProtoNCE}, \tag{10}$$

where $\mathcal{L}_{pred}$ is the loss of the baseline prediction network, λ is a coefficient on the PCL loss term. For easy samples that the network has already fitted perfectly, the PCL loss would hardly bring more benefit in network optimization. Thus, we make λ vary across samples, which performs as a gate to shut off the PCL loss on easy samples. We use the prediction loss $\mathcal{L}_{pred}$ of the network after a warm-up training stage to indicate the hardness of the samples, which is denoted as $\mathcal{L}'_{pred}$. λ is determined according to $\mathcal{L}'_{pred}$:

$$\begin{aligned} \lambda &= a \qquad \mathcal{L}'_{pred} > \theta, \\ \lambda &= 0 \qquad \mathcal{L}'_{pred} < \theta, \end{aligned} \tag{11}$$

where a is a constant hyperparameter, and θ is the threshold to filter out head samples.

4. Experiments

4.1. Datasets

We evaluate our proposed method on several widely used public pedestrain datasets including ETH-UCY, Nuscenes and SDD. ETH-UCY is a pedestrian dataset with rich social interactions. Nuscenes is a large scale trajectory dataset with both vehicles and pedestrians. In this work, we mainly evaluate the performances of our model on the vehicle type, same as [28]. SDD is another large scale bird view trajectory dataset. We use ETH-UCY and Nuscenes in the way same as our backbone Traj++ EWTA [28] and SDD in the way same as our backbone Y-Net [30].

4.2. Evaluation Metrics

Performance metrics. We use the common metrics for evaluating multimodal trajectory prediction performance: Average-Displacement-Error (ADE) and Final-Displacement-Error (FDE), which is commonly used in studies [1, 5, 48]. ADE means the averaged L2 distance between future prediction and ground truth trajectory, while FDE means the L2 distance between the final predicted destination and the ground truth destination. For evaluating multi-modality, we calculate mininum ADE and FDE among all the output guesses, which are denoted as minADE and minFDE and are averaged across the dataset.

Tailed test sample selecting. In order to demonstrate our model on the long-tailed data, we need to separate the hard samples as well as the easy ones for evaluation. Specifically, we use the testing FDEs of the baseline method as the threshold to divide the datasets into seven kinds of samples: the top 1%-5% challenging samples with the largest errors, the rest easier samples, as well as all samples in the datasets. In [28], the Kalman predictor prediction error is utilized for dataset division. Compared with the FDEs of a simple Kalman predictor, performances of an advanced baseline predictor can better reflect the degrees of difficulty for the samples to be modeled by the data-driven network, which can better reveal the ability of the long-tail prediction methods to deal with the hard tailed samples. The Kalman divisions are discussed in supplementaries.

4.3. Baseline

We use Trajectron++ EWTA (Traj++ EWTA) [28] as a baseline for our framework on ETH-UCY and Nuscenes, which has achieved state-of-the-art results according to [28]. Traj++ EWTA augments the Trajectron++ [38] by removing the conditional variational autoencoder parts and using a multi-head decoder trained with the evolving winner-take-all (EWTA) strategy. Another strong baseline we experiment on is Y-Net [30], which uses a U-Net backbone and achieves state-of-the-art results on SDD.

	Top 1%	Top 2%	Top 3%	Top 4%	Top 5%	Rest	All
Traj++ EWTA [28]	0.98/2.54	0.79/2.07	0.71/1.81	0.65/1.63	0.60/1.50	**0.14/0.26**	0.17/0.32
Traj++ EWTA+resample [41]	0.90/2.17	0.77/1.90	0.73/1.78	0.66/1.60	0.64/1.52	0.20/0.41	0.23/0.47
Traj++ EWTA+reweighting [9]	0.97/2.47	0.78/2.03	0.68/1.73	0.62/1.55	0.56/1.40	0.14/0.26	**0.16/0.32**
Traj++ EWTA+LDAM [3]	0.92/2.35	0.76/1.96	0.68/1.71	0.62/1.53	0.57/1.37	0.15/0.27	0.17/0.32
Traj++ EWTA+contrastive [28]	0.92/2.33	0.74/1.91	0.67/1.71	0.60/1.48	0.55/1.32	0.15/0.27	0.17/0.32
Traj++ EWTA+FEND (ours)	**0.84/2.13**	**0.68/1.68**	**0.61/1.46**	**0.56/1.30**	**0.52/1.19**	0.15/0.27	0.17/0.32

Table 1. Prediction errors in the format of (minADE/minFDE) in meters on seven kinds of testing samples on the ETH-UCY dataset.

	Top 1%	Top 2%	Top 3 %	Top 4%	Top 5%	Rest	All
Traj++ EWTA [28]	1.33/3.09	1.02/2.35	0.87/2.00	0.80/1.80	0.74/1.64	0.16/0.26	0.19/0.32
Traj++ EWTA+contrastive [28]	1.28/2.85	0.97/2.15	0.83/1.83	0.76/1.64	0.70/1.48	0.15/0.24	0.18/0.30
Traj++ EWTA w/o resampling+FEND	**1.21/2.50**	**0.92/1.88**	**0.79/1.61**	**0.72/1.43**	**0.66/1.31**	**0.14/0.20**	**0.17/0.26**

Table 2. Prediction errors in the format of (minADE/minFDE) in meters on seven kinds of testing samples on Nuscenes dataset.

4.4. Implement Details

We follow the train schedule of Traj++ EWTA, to train the network with a batch size of 256 for 100 epochs for ETH-UCY and 5 epochs for Nuscenes in each EWTA stage. The learning rate is initially set as 0.01 and exponentially decays with the rate of 0.001. We use a warm-up of 300 epochs in our final model for ETH-UCY. We set $a = 50$ as an initial loss factor same as [28], and a will decade to 0.2 after 100 epochs to not to harm the prediction training process, according to the drop on the EWTA loss. The head sample filter threshold θ is set to 0.2. For the feature extractor, we use a 1D CNN with 16 output channel and a kernel size of 3, attached with an LSTM with a hidden size of 128. For Kmeans clustering, we use $\{20, 50, 100\}$ as the cluster numbers for getting hierarchical clusters. And we use a fully-connected multilayer perception with a hidden size of 128 as the hypernetwork. To train Y-Net, we follow [22] to make the encoded feature with shape (C, H, W) average pooled in the spatial dimension to get a C dimensional vector, and perform PCL on it. We set $a = 1$ and no warmup.

4.5. Comparisons with others

Quantitative comparisons on Traj++ EWTA on ETH-UCY. To show the effectiveness of our methods, we select the state-of-the-art method for long-tail trajectory prediction [28], classical data resampling [41] and loss reweighting [9], and a head-tail performance balancing method [3] for comparison. For long-tailed classification methods [3, 9, 41], we construct a classification head after the encoder of Traj++ EWTA to use it to classify the trajectories according to the discretization of Kalman filter errors, same as Makansi *et al.* [28], and the classification loss is trained along with the prediction loss. Table 1 summarizes our experimental results on ETH-UCY using a best-of-20 evaluation [10]. We can see that our method stably outperforms all comparing methods on all the top $1\% - 5\%$ long-tail samples. Specifically, our framework outperforms the second best method: Traj++ EWTA+contrastive [28] by 9.5% on ADE and 8.5% on FDE on the top 1% hardest samples, and maintains the average ADE and FDE nearly stable. The Traj++ EWTA+reweighting [9] performs best on the average ADE/FDE, but its performance gains on tailed samples are relatively little. The Traj++ EWTA+resampling [41] gets more gains on the most tailed samples, but its average ADE/FDE become much worse. Unlike simply doing resampling or loss reweighting, hypernetwork can decouple head samples and tail samples in the parameter space of decoder, therefore achieves better performances.

Quantitative comparisons on Traj++ EWTA on Nuscenes. Comparison results with the previous best long-tail prediction method [28] on Nuscenes are in Table 2. We find out that the resampling operation in the original Traj++ EWTA does not work well with FEND, probably because of causing overfit on hypernetwork. Despite of this, as shown in Table 2, the baseline without resampling can achieve both superior long-tail and overall performances with FEND. The performances of Traj++ EWTA and Traj++ EWTA+contrastive on both ETH-UCY and Nuscenes are tested on the provided pre-trained models of [28].

Quantitative comparisons on Y-Net on SDD. We also plug our module into Y-Net, the results are shown in Table 3. We reproduced the results of Y-Net using the official released code of [30] with 42 as the random seed, since the original method does not have a fix seed. Results show that our method can achieve performance gains on both tail samples and the whole dataset.

Qualitative comparison. Figure 3 shows some long-tailed hard-case studies of our method on ETH-UCY. Those cases contain some rare social interactions, and all the future trajectories in them are non-trivial to be predicted. In all those samples, our method (blue) outperforms the origi-

	Top 1%	Top 2%	Top 3%	Top 4%	Top 5%	Rest	All
Y-Net* [30]	65.82/134.01	51.84/104.37	43.74/88.21	38.68/76.08	34.72/67.46	**6.54/8.96**	7.93/11.88
Y-Net*+FEND	**57.58/108.51**	**46.33/86.93**	**39.22/75.02**	**35.05/66.24**	**31.27/57.98**	6.64/9.24	**7.87/11.68**

Table 3. Prediction errors in the format of (minADE/minFDE) on seven kinds of testing samples on SDD dataset. * means the results are reproduced using the official released code of [30].

Components			Performance(ADE/FDE)						
PCL	F	H	Top 1%	Top 2%	Top 3%	Top 4%	Top 5%	Rest	All
			0.98/2.54	0.79/2.07	0.71/1.81	0.65/1.63	0.60/1.50	**0.14/0.26**	**0.17/0.32**
✓			0.96/2.41	0.79/2.03	0.70/1.77	0.62/1.56	0.57/1.41	0.15/0.27	0.17/0.32
✓	✓		0.89/2.23	0.72/1.84	0.66/1.61	0.60/1.44	0.55/1.30	0.15/0.27	0.17/0.32
✓		✓	0.90/2.28	0.72/1.87	0.65/1.61	0.58/1.43	0.54/1.30	0.15/0.27	0.17/0.32
✓	✓	✓	**0.84/2.13**	**0.68/1.68**	**0.61/1.46**	**0.56/1.30**	**0.52/1.19**	0.15/0.27	0.17/0.32

Table 4. Ablation study of different modules in FEND. F means future enhanced clusters, H means the hypernetwork.

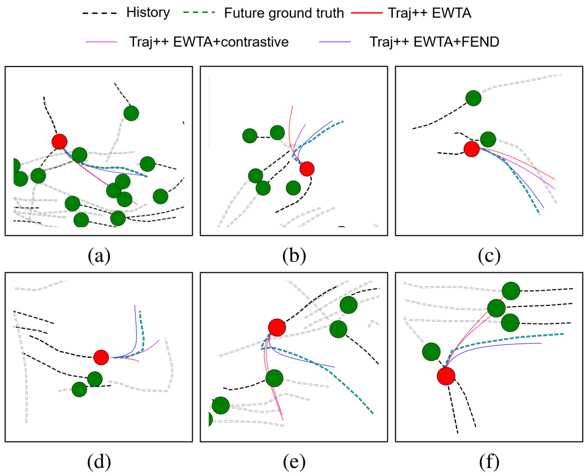


Figure 3. Qualitative results on the ETH-UCY dataset: (a)(b) collision avoidance (c)(d) social influence of parallel walking (e)(f) crowd avoidance. The predictions are selected using a best-of-20 evaluation.

nal Traj++ EWTA (red) and the Traj++ EWTA+contrastive (magenta), thanks to our future enhanced PCL framework for letting the prediction network better recognize different trajectory patterns and a more flexible hyper predictor.

4.6. Ablation Study and Dicussions

Quantitative ablation studies. Results of quantitative ablation studies are shown in Tab. 4. We can see from the results that both the future enhanced clustering and the PCL loss can contribute to the performance of the tailed samples. Importing the hypernetwork can also lead to a decline on the tailed FDEs. And the future enhanced PCL and the hypernetwork are compatible with each other for achieving lower tail FDEs by using both.

Qualitative ablation studies. Figure 4 shows some visualizations of the predict results with our different model variants. All the plotted cases are challenging, and we can see that our full model FEND stably outperforms the other variants and the baseline Traj++ EWTA. Also we can discover from the figure that all of our different model variants perform better than the baseline Traj++ EWTA.

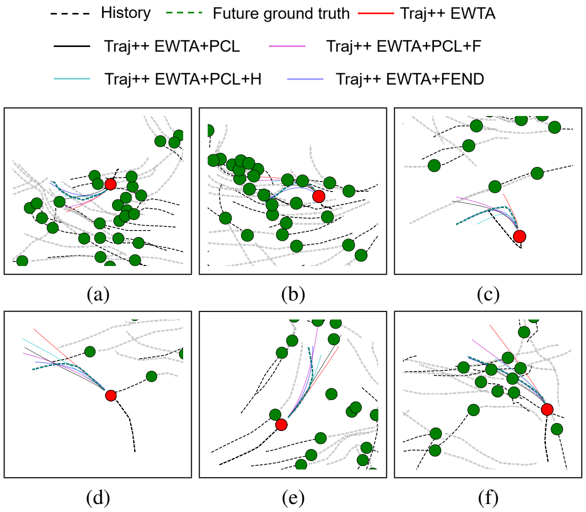
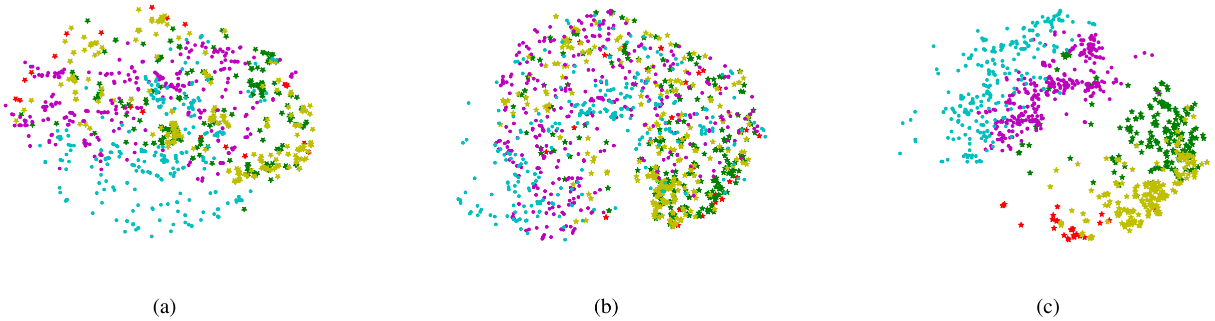


Figure 4. Qualitative results on the ETH-UCY dataset for our different model variants and the baseline Traj++ EWTA. The predictions are selected using a best-of-20 evaluation.

a	Top 1%	Top 2%	Top 3%	All
1	0.97/2.48	0.78/2.01	0.69/1.72	0.17/0.33
20	0.85/2.15	0.68/1.70	0.61/1.47	0.17/0.32
50	**0.84/2.13**	**0.68/1.68**	**0.61/1.46**	**0.17/0.32**
100	0.85/2.14	0.68/1.69	0.61/1.46	0.17/0.32

Table 5. Study on the parameter sensitivity of the auxiliary loss weight a. Results are in the format of (minADE/minFDE) in meters.


(a)

(b)

(c)

Figure 5. TSNE results of (a)Traj++EWTA (b)Traj++ EWTA+contrastive (c)Traj++ EWTA+FEND on the univ scene. The red stars, the green stars, and the yellow stars represent clusters of three kinds of hard tailed patterns, while the magenta and cyan dots represent clusters of two kinds of easy head patterns. We can see from the figures that our method forms a more separately clustered feature space.

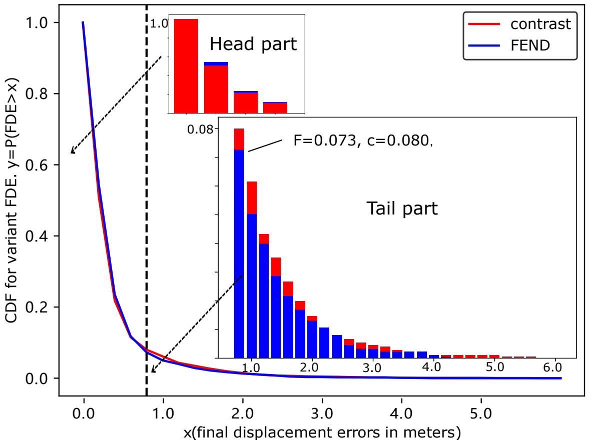


Figure 6. CDF curve and CDF bars of testing FDEs on ETH-UCY. It can be seen that our method have a shorter tail region.

Parameter sensitivity study. Table 5 shows the parameter sensitivity study of PCL loss weight a. We can see that setting $a = 50$ initially will be the best choice. Other parameter sensitivity studies are provided in supplementaries.

Shaped feature embedding space. Figure 5 shows the TSNE results of the feature space of our method and two comparing methods, with two head patterns and three tailed patterns. We can see from the figure that our future enhanced PCL method can decently separate the tail patterns and the head patterns, while there is still some overlap between the heads and the tails in the feature space of Traj++ EWTA and Traj++ EWTA+contrastive. Also, we can see from Fig. 5 that our method can form different clusters for different tailed patterns, while in the feature space of Traj++ EWTA+contrastive, all the samples of the three tail patterns are pushed together, as in Sec. 2 discussed.

FDE distribution bars. To illustrate the distribution of the prediction errors across the dataset more clearly, We plot the cumulative distribution function (CDF) curve of FDEs, and the CDF bars of head and tail regions on ETH-UCY in Figure 6 versus the second-best: Traj++ EWTA+contrastive [28]. The CDF is averaged across the five scenes.

Limitations. The performances on the head samples are slightly dropped, which can been seen in Figure 6 and Table 1 2 3. We leave it as future works. In most experiments we use the minADE/FDE as the prediction evaluation protocols. There are many better metrics such as the Negative Log-Likelihood (NLL) [2, 16, 17] or those which take scene-compliance or socially acceptable prediction into account [18, 21]. The results of another evaluation protocol: FDE NLL are in supplementaries.

Discussion about single agent clustering. We use single agent full trajectory features for clustering, similar to other works using single trajectories to cluster or retrieve [42, 51]. In our experiment we find out that the information in single agent trajectories can already lead to good performances. We believe that it is a promising future direction to include social features into the clustering process.

5. Conclusion

In this paper, we propose a future enhanced contrastive feature space shaping method and a distribution-aware hyper decoder for long-tailed trajectory prediction. Quantitive and qualitative experiment results show that our method can outperform state-of-the-art long-tail prediction methods on the challenging tailed samples, while maintaining the averaged performance on the whole datasets. Our method can be generally plugged into many strong prediction networks.

Acknowledgement

This work was supported by NSFC Projects (No. 62036008) and STI 2030—Major Projects (No. 2021ZD0201300).

References

[1] Alexandre Alahi, Kratarth Goel, Vignesh Ramanathan, Alexandre Robicquet, Li Fei-Fei, and Silvio Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 961–971, 2016. 1, 2, 5

[2] Apratim Bhattacharyya, Bernt Schiele, and Mario Fritz. Accurate and diverse sampling of sequences based on a "best of many" sample objective. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8485–8493, 2018. 8

[3] 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, 6

[4] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924, 2020. 2

[5] Yuning Chai, Benjamin Sapp, Mayank Bansal, and Dragomir Anguelov. Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. *arXiv preprint arXiv:1910.05449*, 2019. 2, 5

[6] Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research*, 16:321–357, 2002. 2

[7] Guangyi Chen, Junlong Li, Nuoxing Zhou, Liangliang Ren, and Jiwen Lu. Personalized trajectory prediction via distribution discrimination. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15580–15589, 2021. 2

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 2

[9] Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9268–9277, 2019. 2, 6

[10] Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. Social gan: Socially acceptable trajectories with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2255–2264, 2018. 1, 6

[11] David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016. 2, 4

[12] Marah Halawa, Olaf Hellwich, and Pia Bideau. Action-based contrastive learning for trajectory prediction. In *European Conference on Computer Vision*, pages 143–159. Springer, 2022. 2

[13] Hui Han, Wen-Yuan Wang, and Bing-Huan Mao. Borderline-smote: a new over-sampling method in imbalanced data sets learning. In *International conference on intelligent computing*, pages 878–887. Springer, 2005. 2

[14] John A Hartigan and Manchek A Wong. Algorithm as 136 a k-means clustering algorithm. *Journal of the royal statistical society. series c (applied statistics)*, 28(1):100–108, 1979. 3

[15] Haibo He and Edwardo A Garcia. Learning from imbalanced data. *IEEE Transactions on knowledge and data engineering*, 21(9):1263–1284, 2009. 2

[16] Ronny Hug, Wolfgang Hübner, and Michael Arens. Introducing probabilistic bézier curves for n-step sequence prediction. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 10162–10169, 2020. 8

[17] Boris Ivanovic and Marco Pavone. The trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2375–2384, 2019. 8

[18] Boris Ivanovic and Marco Pavone. Injecting planning-awareness into prediction and detection evaluation. In *2022 IEEE Intelligent Vehicles Symposium (IV)*, pages 821–828. IEEE, 2022. 8

[19] Yannis Kalantidis, Mert Bulent Sariyildiz, Noe Pion, Philippe Weinzaepfel, and Diane Larlus. Hard negative mixing for contrastive learning. *Advances in Neural Information Processing Systems*, 33:21798–21809, 2020. 2

[20] Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in Neural Information Processing Systems*, 33:18661–18673, 2020. 2

[21] Parth Kothari, Sven Kreiss, and Alexandre Alahi. Human trajectory forecasting in crowds: A deep learning perspective. *IEEE Transactions on Intelligent Transportation Systems*, 23(7):7386–7400, 2021. 8

[22] Mihee Lee, Samuel S Sohn, Seonghyeon Moon, Sejong Yoon, Mubbasir Kapadia, and Vladimir Pavlovic. Muse-vae: multi-scale vae for environment-aware long term trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2221–2230, 2022. 6

[23] Junnan Li, Pan Zhou, Caiming Xiong, and Steven CH Hoi. Prototypical contrastive learning of unsupervised representations. *arXiv preprint arXiv:2005.04966*, 2020. 2, 3, 4

[24] Tianhong Li, Peng Cao, Yuan Yuan, Lijie Fan, Yuzhe Yang, Rogerio S Feris, Piotr Indyk, and Dina Katabi. Targeted supervised contrastive learning for long-tailed recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6918–6928, 2022. 1

[25] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 2

[26] Yuejiang Liu, Qi Yan, and Alexandre Alahi. Social nce: Contrastive learning of socially-aware motion representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15118–15129, 2021. 2

[27] Yuanfu Luo, Panpan Cai, Aniket Bera, David Hsu, Wee Sun Lee, and Dinesh Manocha. Porca: Modeling and planning for autonomous driving among many pedestrians. *IEEE Robotics and Automation Letters*, 3(4):3418–3425, 2018. 1

[28] 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. 1, 2, 4, 5, 6, 8

[29] Osama Makansi, Eddy Ilg, Ozgun Cicek, and Thomas Brox. Overcoming limitations of mixture density networks: A sampling and fitting framework for multimodal future prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7144–7153, 2019. 2

[30] Karttikeya Mangalam, Yang An, Harshayu Girase, and Jitendra Malik. From goals, waypoints & paths to long term human trajectory forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15233–15242, 2021. 2, 5, 6, 7

[31] Aditya Krishna Menon, Sadeep Jayasumana, Ankit Singh Rawat, Himanshu Jain, Andreas Veit, and Sanjiv Kumar. Long-tail learning via logit adjustment. *arXiv preprint arXiv2007.07314*, 2020. 2

[32] 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. 2

[33] Sriram Narayanan, Ramin Moslemi, Francesco Pittaluga, Buyu Liu, and Manmohan Chandraker. Divide-and-conquer for lane-aware diverse trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15799–15808, 2021. 2

[34] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 2

[35] Bo Pang, Tianyang Zhao, Xu Xie, and Ying Nian Wu. Trajectory prediction with latent belief energy-based model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11814–11824, 2021. 2

[36] Tung Phan-Minh, Elena Corina Grigore, Freddy A Boulton, Oscar Beijbom, and Eric M Wolff. Covernet: Multimodal behavior prediction using trajectory sets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14074–14083, 2020. 2

[37] Amir Sadeghian, Vineet Kosaraju, Ali Sadeghian, Noriaki Hirose, Hamid Rezatofighi, and Silvio Savarese. Sophie: An attentive gan for predicting paths compliant to social and physical constraints. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1349–1358, 2019. 1

[38] 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, 2, 4, 5

[39] Dvir Samuel and Gal Chechik. Distributional robustness loss for long-tail learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9495–9504, 2021. 1

[40] Nasim Shafiee, Taskin Padir, and Ehsan Elhamifar. Introvert: Human trajectory prediction via conditional 3d attention. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16815–16825, 2021. 2

[41] Li Shen, Zhouchen Lin, and Qingming Huang. Relay backpropagation for effective learning of deep convolutional neural networks. In *European conference on computer vision*, pages 467–482. Springer, 2016. 2, 6

[42] Jianhua Sun, Yuxuan Li, Hao-Shu Fang, and Cewu Lu. Three steps to multimodal trajectory prediction: Modality clustering, classification and synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 13250–13259, 2021. 2, 8

[43] Chenxin Xu, Maosen Li, Zhenyang Ni, Ya Zhang, and Siheng Chen. Groupnet: Multiscale hypergraph neural networks for trajectory prediction with relational reasoning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6498–6507, 2022. 2

[44] Chenxin Xu, Weibo Mao, Wenjun Zhang, and Siheng Chen. Remember intentions: Retrospective-memory-based trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6488–6497, 2022. 2

[45] Yuzhe Yang, Kaiwen Zha, Yingcong Chen, Hao Wang, and Dina Katabi. Delving into deep imbalanced regression. In *International Conference on Machine Learning*, pages 11842–11851. PMLR, 2021. 2

[46] Cunjun Yu, Xiao Ma, Jiawei Ren, Haiyu Zhao, and Shuai Yi. Spatio-temporal graph transformer networks for pedestrian trajectory prediction. In *European Conference on Computer Vision*, pages 507–523. Springer, 2020. 2

[47] Ye Yuan, Xinshuo Weng, Yanglan Ou, and Kris M Kitani. Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9813–9823, 2021. 2

[48] Pu Zhang, Wanli Ouyang, Pengfei Zhang, Jianru Xue, and Nanning Zheng. Sr-lstm: State refinement for lstm towards pedestrian trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12085–12094, 2019. 1, 2, 5

[49] Pu Zhang, Jianru Xue, Pengfei Zhang, Nanning Zheng, and Wanli Ouyang. Social-aware pedestrian trajectory prediction via states refinement lstm. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 1, 4

[50] Hang Zhao, Jiyang Gao, Tian Lan, Chen Sun, Ben Sapp, Balakrishnan Varadarajan, Yue Shen, Yi Shen, Yuning Chai, Cordelia Schmid, et al. Tnt: Target-driven trajectory prediction. In *Conference on Robot Learning*, pages 895–904. PMLR, 2021. 2

[51] He Zhao and Richard P Wildes. Where are you heading? dynamic trajectory prediction with expert goal examples. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 7629–7638, 2021. 8