PreTraM: Self-Supervised Pre-training via Connecting Trajectory and Map

Chenfeng Xu*[1], Tian Li*[2], Chen Tang**[1], Lingfeng Sun[1], Kurt Keutzer[1], Masayoshi Tomizuka[1], Alireza Fathi[3], and Wei Zhan[1]

[1] University of California, Berkeley
[2] University of California, San Diego
[3] Google Research

Abstract. Deep learning has recently achieved significant progress in trajectory forecasting. However, the scarcity of trajectory data inhibits the data-hungry deep-learning models from learning good representations. While pre-training methods for representation learning exist in computer vision and natural language processing, they still require large-scale data. It is hard to replicate their success in trajectory forecasting due to the inadequate trajectory data (e.g., 34K samples in the nuScenes dataset). To work around the scarcity of trajectory data, we resort to another data modality closely related to trajectories—HD-maps, which is abundantly provided in existing datasets. In this paper, we propose *PreTraM*, a self-supervised **Pre**-training scheme via connecting **Tra**jectories and **M**aps for trajectory forecasting. PreTraM consists of two parts: 1) Trajectory-Map Contrastive Learning, where we project trajectories and maps to a shared embedding space with cross-modal contrastive learning, 2) Map Contrastive Learning, where we enhance map representation with contrastive learning on large quantities of HD-maps. On top of popular baselines such as AgentFormer and Trajectron++, PreTraM reduces their errors by 5.5% and 6.9% relatively on the nuScenes dataset. We show that PreTraM improves data efficiency and scales well with model size. Our code and pre-trained models will be released at https://github.com/chenfengxu714/PreTraM.

Keywords: Trajectory Forecasting, Self-Supervised Learning, Pre-training, Contrastive Learning, Multi-modality

1 Introduction

Trajectory forecasting is a challenging task in autonomous driving, which aims at predicting the future trajectory conditioned on past trajectories and surrounding scenes. Current deep learning models have dominated trajectory forecasting by data-driven supervised learning. However, both the collection and the annotation of trajectory data are extremely difficult and costly. Trajectory data is collected

* Equal contribution
** Corresponding author

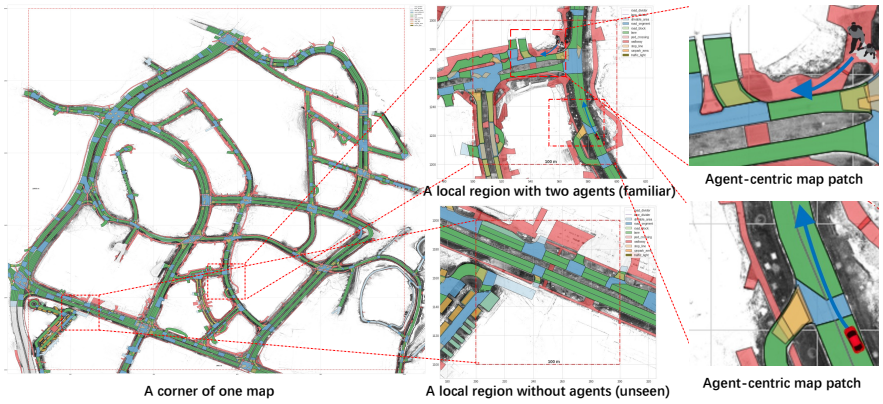

Fig. 1. We have two key observation about maps and trajectories: 1) As shown in the rightmost column, vehicles usually move in drivable areas and pedestrians usually move along sidewalks. And the relationship learnt from familiar scenes can generalize to unseen scenes. (Please zoom in for better view.) 2) Agent-centric map patches are taken from a local region of the map, which is just a tiny part of the whole map.

by vehicles with sophisticated sensor systems. Then annotators need to label the objects, associate their positions, generate and smoothen trajectories. This complex procedure limits the scale of the data. The popular open-sourced trajectory forecasting dataset nuScenes [3] has only 34K samples, much fewer than that of the elementary small-scale image dataset MNIST (60K samples) [10]. The scarcity of trajectory data prohibits the models from learning good trajectory representation, and thus restrains their performance.

In the Natural Language Processing (NLP) and computer vision (CV) communities, it was found effective to use self-supervised pre-training on vast unlabeled datasets to learn language/visual representations. The classic methods, such as autoregressive language modeling [2], masked autoencoding [12], and contrastive learning [6, 18], are conceptually simple, but require billions of training data. Although recent results from CLIP [28] show that cross-modal contrastive learning requires much fewer pre-training data (4x fewer), the amount of data used is still far more than available trajectory data. Unlike NLP and CV, where large-scale unlabeled datasets exist, the bottleneck for scaling trajectory datasets lies in data collection and annotation. It poses the critical challenge for trajectory forecasting to benefit from existing pre-training schemes. And to the best of our knowledge, few efforts in trajectory forecasting have explored pre-training.

To work around the scarcity of trajectories, we resort to another modality of data that is closely related to trajectories—HD-maps. In fact, we observe two important facts about maps:

– An agent's trajectory is correlated to the map around it [13, 24]. A representative example is that the shape of trajectory usually follows the topology

of the HD-map. As shown in the rightmost column of Figure 1, vehicles usually move in drivable areas, and pedestrians usually move along sidewalks. More importantly, the relationships between trajectory and map can be generalized to other scenes. For example, in the middle of Figure 1, the model learns from the upper scene that the moving car should *follow the boundary of the road*. By capturing this relationship, the model knows that a car in the unseen bottom scene should also *follow the boundary of the road*.
– Existing works in trajectory forecasting only take advantage of the agent-centric map patches, the local regions containing at least one annotated trajectory, but significantly under-utilize other parts of the maps, which cover much larger areas. As shown in Figure 1, agent-centric map patches are tiny compared with the leftmost global map.

Based on the above observations, we propose PreTraM, a self-supervised **pre**-training scheme via connecting **tra**jectories and **m**aps for trajectory forecasting. Specifically, we jointly pre-train the trajectory encoder and the map encoder of a model in two ways: 1) *Trajectory-Map Contrastive Learning (TMCL)*: Inspired by CLIP [28], we constrast trajectories with corresponding map patches to enforce the model to capture their relationship. 2) *Map Contrastive Learning (MCL)*: We train a stronger map encoder with contrastive learning on large quantities of trajectory-decoupled map patches, which outnumber the agent-centric ones by 782x. In short, PreTraM is a synergy of TMCL and MCL: a better trajectory representation is learned via bridging the map and trajectory representations with TMCL, so that the trajectory encoder benefits from the map representation enhanced by MCL.

Our method reduces the prediction error of a variety of popular prediction models including AgentFormer [35] and Trajectron++ [29] by 5.5% and 6.9% relatively on the nuScenes dataset [3]. More importantly, we find that PreTraM is able to achieve larger performance gain when less data is available. Impressively, using only 70 % of the trajectory data, PreTraM on top of AgentFormer show superior performance than AgentFormer trained on 100 % trajectory data. This demonstrates the proposed pre-training scheme brings strong data efficiency. Furthermore, we apply PreTraM to larger versions of AgentFormer and observe it consistently improves prediction accuracy when the model scales up. We also conduct sufficient ablation studies and shed light on how PreTraM works.

In summary, our key contributions are as follows:

– We propose PreTraM, a novel self-supervised pre-training scheme for trajectory forecasting by connecting trajectories and maps, which consists of trajectory-map contrastive learning and map contrastive learning.
– We show with experiments that PreTraM achieves up to 6.9 % relative improvement in FDE-10 upon popular baselines.
– PreTraM enhances the data efficiency of prediction models, using 70% training data but beating the baseline with 100% training data, and generalizes to models of larger scales.
– Through ablation studies and analysis, we demonstrate the efficacy of TMCL and MCL respectively, and shed light on how PreTraM works.

2 Background

2.1 Problem Formulation of Trajectory Forecasting

In trajectory forecasting, we aim to predict the future trajectories of multiple target agents in a scene. Typically, a set of history states x for all agents and the surrounding HD-map patches M are input to the model f_ω and the model predicts the future trajectories of each agent $y = f_\omega(x, M)$.

The HD-map contains rich semantic information (e.g., drivable area, stop line, and traffic light) [3]. In this work, we employ rasterized top-down semantic images around each of the agents as the input HD-map patches M, i.e., $M = \{m_i\}_{i\in\{1,\dots,A\}}, m_i \in \mathbb{R}^{C\times C\times 3}$, where C is the context size and 3 denotes the RGB channels. Note that each color has its specific semantic meaning in HD-maps.

As for the history states, denoting the number of agents in the scene as A, and the history time span as T, then $x = s_{1,\dots,A}^{(-T:0)} \in \mathbb{R}^{T\times A\times D}$, where s_i is the history states of agent i, and 0 denotes the current timestamp. D is the dimension of features that generally contain the agent's 2D or 3D coordinates, as well as other information such as its heading and its speed.

2.2 Contrastive Learning

Contrastive learning is a powerful method for self-supervised representation learning that was made popular by [6–8,18]. Using instance discrimination as the pretext task, they pull the semantically-close neighbors together and push away non-neighbors [14]. For example, in SimCLR [6], given a mini-batch of inputs, each input x_i is transformed into a positive sample x_i^+. Let h_i, h_i^+ denote the hidden representation of x_i, x_i^+. Then on a mini-batch of N pairs of (x_i, x_i^+), it adopts the InfoNCE loss [25] as its training objective.

In particular, we are interested in one specific work that explored contrastive learning in NLP: SimCSE [14]. Instead of using word replacement or deletion as augmentation, it uses different dropout masks in the model as the minimal augmentation for the positive samples. This simple approach turns out to be very effective in that it fully preserves the semantic of the text, compared with other augmentation operations. To preserve the semantic of HD-map, we also adopt dropout for the positive samples in map contrastive learning.

More recently, CLIP [28] demonstrated the power of cross-modal contrastive learning conditioned on huge amounts of data. It collects paired images and captions from the Internet and asks the model to pair an image with the corresponding text, using large batches. For a mini-batch of N pairs of images I_i and texts T_i, denoting their hidden representations as (h_i^I, h_i^T), it applies cross-entropy loss on the $N \times N$ similarity matrix over all pairs of images and texts, stated as follows:

$$l_i = -\log \frac{e^{\mathrm{sim}(h_i^I, h_i^T)/\tau}}{\sum_{j=0}^{N} e^{\mathrm{sim}(h_i^I, h_j^T)/\tau}} \tag{1}$$

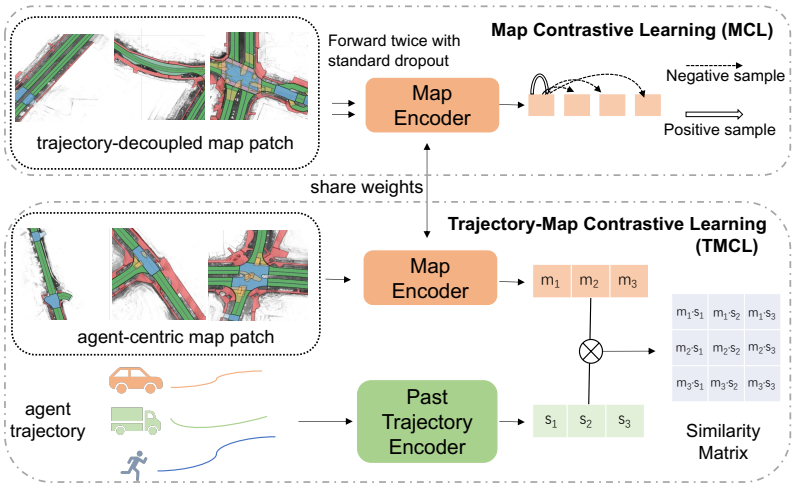

Fig. 2. Top: Map Contrastive Learning (MCL). On the contrary to agent-centric map patches, the trajectory-decoupled ones do not necessarily contain agent trajectories. During training, we randomly crop those patches from the whole map around positions on the road. Bottom: Trajectory-Map Contrastive Learning (TMCL).

where $\text{sim}(\cdot, \cdot)$ is a measurement of similarity, typically the cosine similarity, and τ is the temperature parameter. Note that it can be seen as the InfoNCE loss using the corresponding text as the positive sample of an image.

Intuitively, using natural language as supervision of images, CLIP puts image and text in a shared embedding space. Besides, equation (1) enforces similarity between the correct pair of images and text, and thus learns the pattern of image-text relationship. Following this intuition, we design a trajectory-map contrastive learning objective to capture the relationship between them.

3 Method

We propose a novel self-supervised **pre**-training scheme by connecting **tra**jectory and **m**ap (PreTraM) to enhance the trajectory and map representations when there are small-scale trajectory data, but large-scale map data. We jointly pre-train a trajectory encoder and a map encoder to obtain good trajectory representation by encoding the trajectory-map relationship into the representation.

As illustrated in Figure 2, the proposed PreTraM is composed of two parts: 1) A simple trajectory-map contrastive learning (TMCL) that is conducted between map encoder and trajectory encoder, using limited number of trajectories and the paired map patches. 2) A simple map contrastive learning (MCL) that is conducted on map encoder using large batch size on trajectory-decoupled map patches, where there are not necessarily agent trajectories. After pre-training, we load the pre-trained weights and finetune under the prediction objective with the same training schedules as the original models.

3.1 Trajectory-Map Contrastive Learning (TMCL)

We propose to use a cross-modal contrastive learning method that facilitates both trajectory encoder and map encoder. Specifically, given a mini-batch of scenes, for all the input history states x, we split them into single agent trajectories and treat them *independently*, i.e., $S = \{s_i | s_i \in x, \forall x \in B\}$, where B denotes the mini-batch. For each agent, we crop an agent-centric map patch around its current position from the HD-map, and then we have $N_{\text{traj}} = |S|$ pairs of correlated trajectories and HD-map patches (s_i, m_i). We also rotate the map with respect to the orientation of the agent following the common practice [29, 35]. The model is required to match each trajectory s_i with the paired map patch m_i among all the map patches in the mini-batch and vice versa. As shown in bottom of Figure 2, we input the trajectories and maps into the corresponding encoders to obtain the features $\{h_i^{\text{traj}}\}, \{h_i^{\text{map}}\}$. Then we compute a similarity matrix across all pairs of trajectories and maps in the mini-batch. Note that we apply a linear projection layer [28] after the map encoder and the trajectory encoder for the hidden representations h but omit it above for the sake of simplicity. It is the same case in section 3.2 for MCL. Finally, we optimize a symmetric cross-entropy loss over these similarity scores as follows [28]:

$$l_i^{\text{TMCL}} = -\log \frac{e^{\text{sim}(h_i^{\text{traj}}, h_i^{\text{map}})/\tau_{\text{traj}}}}{\sum_{j=1}^{N_{\text{traj}}} e^{\text{sim}(h_i^{\text{traj}}, h_j^{\text{map}})/\tau_{\text{traj}}}} \tag{2}$$

Through this objective, the similarities of the correct pairs of the trajectories and maps are maximized and those of the other pairs are minimized. It results in a shared embedding space of trajectories and maps. We find that a prediction model that fuses map and trajectories to make prediction benefits from such a shared embedding space. It agrees with the finding in [20] that models for vision-language tasks benefit from an aligned embedding space for the visual and language inputs. The TMCL objective teaches the model to encode the relationship between maps and trajectories into the representation. By capturing the relationship, the trajectory embedding contains the information of the underlying map conditioned on the input trajectory, which implies the geometric and routing information of the future trajectories for the predictor.

3.2 Map Contrastive Learning (MCL)

To further facilitate learning the trajectory-map relationship, we learn a general map representation by map contrastive learning. At each training iteration, we randomly crop N_{map} map patches from a random subset of HD-Maps in the dataset. Note that N_{map} is much greater than the agent number A.

In addition to using a large number of map patches, the key ingredient to get MCL to work effectively is using the exact identical instance as its positive sample, i.e., $m_i^+ = m_i$, and apply dropout in the map encoder [14]. Denote the hidden representation $h_i^z = g_\theta(m_i, z)$ where g_θ is the map encoder and z is a random mask for dropout. As shown in the top part of Figure 2, we feed the the

same map patch to the encoder in two independent forward passes with different dropout masks z, z', which gives two representation $h_i^z, h_i^{z'}$ for each m_i. Thanks to dropout, h_i^z and $h_i^{z'}$ are different, but still encode the same topology and semantic. In contrast, regular augmentation operations in CV such as random rotation, flip, gaussian noise or color jitter do not work here. Gaussian noise and color jitter transform the semantics of HD-maps, while flip and rotation change their topologies. Instead, dropout serves as a minimal augmentation for the positive sample and turns out to be effective through experiments. We provide comparison experiments on dropout against other augmentations in Section 2 of the supplementary material. Formally, the training objective of MCL is:

$$l_i^{\text{MCL}} = -\log \frac{e^{\text{sim}(h_i^{z_i}, h_i^{z'_i})/\tau_{\text{map}}}}{\sum_{j=1}^{N_{\text{map}}} e^{\text{sim}(h_i^{z_i}, h_j^{z'_j})/\tau_{\text{map}}}} \tag{3}$$

It is worth noting that MCL is a novel use of HD-maps not only because we make use of every piece of the HD-map, but also because we design a customized training objective to make better use of HD-maps.

3.3 Training Objective

The PreTraM scheme is complete with the joint of TMCL and MCL. The overall objective function combines their objectives, given by:

$$\mathcal{L} = \sum_{i=1}^{N_{\text{traj}}} l_i^{\text{TMCL}} + \lambda \sum_{i=1}^{N_{\text{map}}} l_i^{\text{MCL}} \tag{4}$$

4 Experiments

4.1 Dataset and implementation details

Dataset. nuScenes is a recent large-scale autonomous driving dataset collected from Boston and Singapore. It consists of 1000 driving scenes with each scene annotated at 2Hz, and the driving routes are carefully chosen to capture challenging scenarios. The nuScenes dataset provides HD semantic maps from Boston Seaport together with Singapore's One North, Queenstown and Holland Village districts, with 11 semantic classes. It is split into 700 scenes for training, 150 scenes for validation, and 150 scenes for testing.

Our main experiments follow the split used in AgentFormer [35], in which the original training set is split into two parts: 500 scenes for training, and 200 scenes for validation. The original validation set is used for testing our model.

Baseline. We performed experiments with PreTraM on two models, AgentFormer [35] and Trajectron++ [29]. Both of them are CVAE models including a past trajectory encoder, a map encoder, a future trajectory encoder, and a future

trajectory decoder. We reproduced AgentFormer to support parallel training. Compared with the original code, our reproduced code trains 17.1x faster than the official code (4.5 hours vs. 77 hours on one V100 GPU), and its performance is competitive—0.029 better than the official implementation on ADE-5. Note that AgentFormer separately trains DLow [34] for better sampling. We did not reproduce this part since we focus on representation learning and want a precise quantitative evaluation on the benefit of PreTraM to the model itself. Plus, Trajectron++ does not use DLow while applicable. We want to keep the setting consistent, so that it is meaningful to compare the performance gains between different models. As for Trajectron++ [29], we use their official implementation but re-train it using the data split in AgentFormer to ensure fair comparison. In the following sections, we denote AgentFormer/Trajectron++ pre-trained with PreTraM as PreTraM-AgentFormer/PreTraM-Trajectron++, or PreTraM when the model is clear in the context.

Pre-training and finetuning. Our pre-training is applied to the *past* trajectory encoder and map encoder. To train TMCL, we pair the historical trajectories of last 2s and map patches of context size 100×100. We randomly rotate the trajectories and maps simultaneously for data augmentation. For MCL, we collect the trajectory-decoupled map patches dynamically at training. For each instance in the mini-batch, we crop 120 map patches centered at random positions along the road in the HD-map. We pre-train the encoders with the PreTraM objective function for 20 epochs using batch size 32 (which means 3440 map patches for MCL in one iteration). Throughout the pre-training phase, we use 28.8M map patches to train our map encoder, which is 782x more than agent-centric map patches. The pre-training phase is fast—only *30 minutes* on one V100 GPU for AgentFormer. More details are in Section 3 of the Supplementary material.

Recall that we use dropout for positive samples in MCL. In shallow map encoders, such as Map-CNN used in AgentFormer and Trajectron++, we place the dropout at post-activation of each convolution. For relatively deeper map encoder such as ResNet family, we place two dropout masks on each residual block. The mask ratio of dropout is default as $p = 0.1$.

At finetuning phase, we use the same training recipes as AgentFormer and Trajectron++. The prediction horizon is 6 seconds and we use the ground-truth future trajectories to supervise the training.

Metric. The main metrics are Average Displacement Error (ADE) and Final Displacement Error (FDE). We follow previous works [29,35] to sample k trajectories during inference and pick the minimum of the error, denoting as ADE-k and FDE-k. Apart from the sampling based metrics above, we also use a deterministic metric meanFDE, which is the FDE of the trajectory that the model deems as the most likely.

We also leverage the metrics including Kernel Density Estimate-based Negative Log Likelihood (KDE NLL) [29] and boundary violation rate. The former measures the NLL of the ground truth trajectory under a distribution created

Table 1. Comparison experiments based on AgentFormer [35] and Trajectron++ [29]. Note that the reported AgentFormer is removed of DLow. The AgentFormer* denotes our reproduced implementation. *Lower* number is better.

Method	ADE-5	FDE-5	ADE-10	FDE-10
MTP [9]	2.93	-	-	-
AgentFormer [35]	2.517	5.459	1.852	3.869
MultiPath [5]	2.32	-	1.96	-
DLow-AF [34]	2.11	4.70	1.78	3.58
DSF-AF [23]	2.06	4.67	1.66	3.71
CoverNet [26]	1.96	-	1.48	-
AgentFormer*	2.488	5.420	1.893	3.902
PreTraM-AgentFormer*	**2.391**(-0.097)	**5.177**(-0.243)	**1.796**(-0.097)	**3.687**(-0.215)
Trajectron++ [29]	1.772	4.150	1.405	3.221
PreTraM-Trajectron++	**1.698**(-0.074)	**3.963**(-0.197)	**1.348**(-0.057)	**3.040**(-0.181)

Table 2. Experimental evaluation on meanFDE, KDE NLL, and Boundary violation rate (B. Viol.) provided by Trajectron++ [29]. *Lower* number is better.

Method	meanFDE	KDE NLL	B. Viol. (%)
Trajectron++	8.242	2.487	23.7
PreTraM-Trajectron++	**8.212**(-0.030)	**2.380**(-0.107)	**21.9**(-1.8)

by fitting a kernel density estimate on trajectory samples, which shows the likelihood of the ground truth trajectory given the sampled trajectory predictions. The latter is the ratio of the predicted trajectories that hit road boundaries.

4.2 Comparison Experiments

The results compared with the baselines and the other prior-arts are shown in Table 1. Observe that using PreTraM improves the performance by 0.097 (*resp. 0.074*) ADE-5, 0.243 (*resp. 0.197*) FDE-5, 0.097 (*resp. 0.057*) ADE-10, 0.215 (*resp. 0.181*) FDE-10, on top of AgentFormer (*resp.* Trajectron++). This is up to *4.1%* relative improvement on ADE-5 and *6.9%* on FDE-10. Remarkably, we achieve it with a simple pre-training scheme. PreTraM does not rely on long pre-training epochs or huge quantities of external data as said in section 4.1. The HD-map we use during pre-training is inherently provided by the dataset. Besides, PreTraM is plug-and-play, and can be easily applied to most prediction model that fuses HD-map and trajectory. In conclusion, these results demonstrate that PreTraM indeed facilitates the models in representation learning. Note that our reproduced AgentFormer does not include DLow as stated in Section 4.1.

We also evaluate the results on the metrics provided by Trajectron++ to show the advantage of PreTraM. PreTraM-Trajectron++ improves baseline by 0.107 KDE NLL and 1.8% boundary violation rate (Table 2). The improvements on these two metrics show that our pre-training scheme not only improves prediction accuracy, but also improves stability and safety.

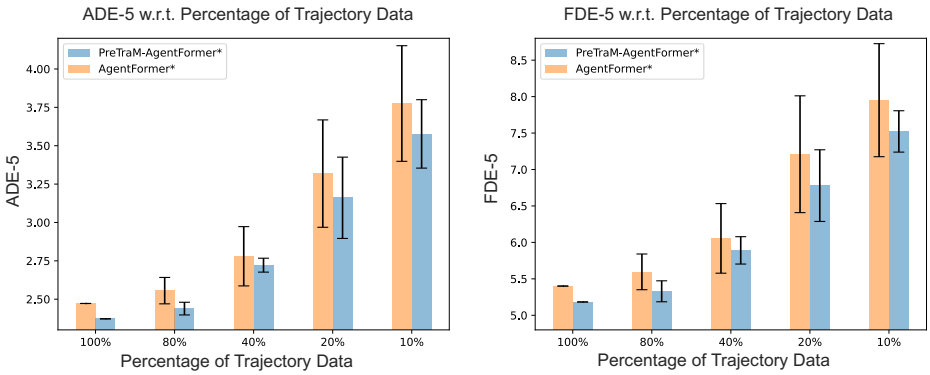

Fig. 3. Experiments with part of the trajectory data. Left: ADE-5 results. Right: FDE-5 results. We repeat the experiments with 3 different random seeds and report the mean performance. The error bars are 3 times the standard deviation. As the percentage of trajectory data becomes lower, the improvements of PreTraM are larger. Moreover, the std of PreTraM is much smaller than the baseline over all the settings.

4.3 Data Efficiency

In this section, we explore whether the learned representations of trajectory and map can improve data efficiency. To investigate this we evaluate PreTraM-AgentFormer on a fraction of the dataset, comparing its result with baseline AgentFormer. In this set of experiments to best demonstrate our strength, we use ResNet18 as a substitute for the 4-layer map encoder, Map-CNN, in the original AgentFormer. This is due to the intuition that larger models are better at representation learning [6, 18, 28]. We randomly sample 80%, 40%, 20% and 10% trajectories from the dataset, but keep all the HD-maps available. For each setting, we repeat the experiments with 3 different random seeds and report the mean and the standard deviation in Figure 3. We observe that PreTraM-AgentFormer outperforms the baseline in all settings. More importantly, the performance gain of PreTraM gets larger as the percentage of data goes smaller. With *10%* of data, i.e., around 1200 samples, PreTraM surpasses the baseline by *0.32* on ADE-5 and *0.77* on FDE-5. Moreover, the std of PreTraM is much smaller than the baseline. The std of ADE-5 of the baseline are 0.035 (80%), 0.079 (40%), 0.143 (20%), and 0.154 (10%) respectively, while those of PreTraM are 0.017, 0.018, 0.108, 0.091. It is the same case in terms of FDE-5.

In addition, we observe that training on 70% of data, PreTraM-AgentFormer still outperforms the baseline with 100 % of data (2.470 ADE-5 vs. 2.472 ADE-5). More results are shown in Section 1 of the supplementary material.

4.4 Scalability Analysis

A good representation learning method is able to scale with the model size [6, 18]. Therefore, we evaluate PreTraM on map encoders and trajectory encoders

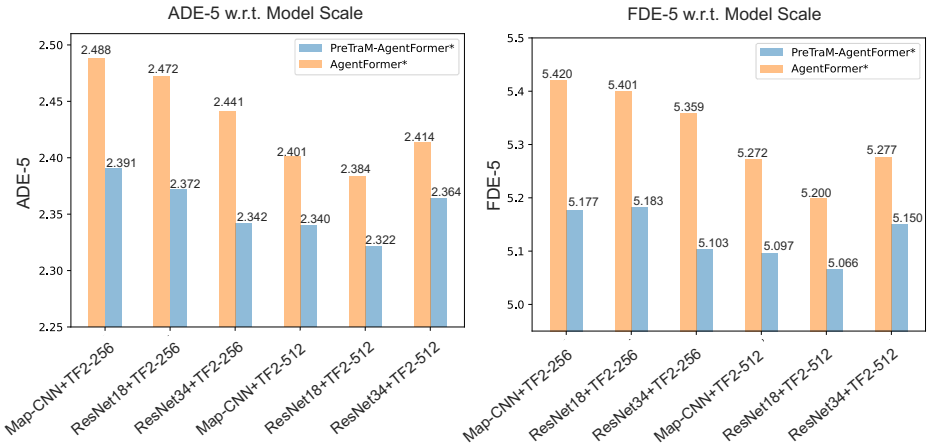

Fig. 4. Experiment with models at different scales. As the model gets deeper and wider, PreTraM consistently improves AgentFormer by a large margin.

of different depth and width. Map-CNN is the map encoder used in original AgentFormer. It is merely a 4-layer convolutional network. Alternatively, we use ResNet18 or ResNet34 as the map encoder. Besides, we tried trajectory encoders of various channel size including 256 and 512. As shown in Figure 4, PreTraM consistently improves ADE and FDE upon models of different scales. Note that we observe overfitting of the ResNet34+TF-512 model, which is why its performance degrades compared with smaller models.

4.5 Analysis

It is natural that PreTraM enhances the map representation since we utilize 28.8M map samples for pre-training the map encoder, but as we proposed in previous sections, another important goal is to further enhance the trajectory representation. Therefore, we conduct experiments and delve deep into the function of PreTraM to discuss how it enhances the trajectory representation. All the experiments are completed upon AgentFormer with Map-CNN.

Does PreTraM indeed improve trajectory representation? In fact, we can quantitatively demonstrate this by loading one of the pre-trained encoders, trajectory encoder (TE) and map encoder (ME), at finetuning phase. As shown in Table 3, we can first observe that loading only the pre-trained map encoder improves prediction performance. More importantly, we observe that just loading TE pre-trained weights is able to give almost the same result as loading both of ME and TE. This means the learnt trajectory representation is strong, and that the major benefit of PreTraM owes to the trajectory representation.

So our answer is "Yes, PreTraM indeed improves trajectory representation."

Is TMCL crucial for improving trajectory representation? To examine the contribution of TMCL, we experiment with an alternative to TMCL

Table 3. Comparison with loading one of the pretrained models when finetuning. ME: map encoder, TE: past trajectory encoder.

Method	ADE-5	FDE-5
Baseline AgentFormer	2.488	5.420
Finetune with both pretrained weights (PreTraM)	2.391(-0.097)	5.177(-0.243)
Finetune with only TE pretrained weights	2.399(-0.089)	5.277(-0.143)
Finetune with only ME pretrained weighst	2.454(-0.034)	5.372(-0.048)

Table 4. Comparison with different pre-training strategies. MTM means masked trajectory modeling, recovering the masked trajectories during pre-training, which is a mimic of masked language modeling in NLP [12].

Method	ADE-5	FDE-5
Baseline AgentFormer	2.488	5.420
Pre-training with both TMCL and MCL (PreTraM)	2.391(-0.097)	5.177(-0.243)
Pre-training with MTM and MCL	2.431(-0.057)	5.322(-0.098)
Pre-training with only MCL	2.442(-0.046)	5.373(-0.057)
Pre-training with only TMCL	2.451(-0.037)	5.369(-0.051)

as the objective function for pre-training trajectory representation. Inspired by Masked Language Modeling (MLM) [12] for sequence modeling in NLP, we randomly mask out part of the input history states and ask the trajectory encoder to recover the masked part. Denoting this task as Masked Trajectory Modeling (MTM), we jointly pre-train the model on the objective of MTM and MCL. For variable controlling, we also pre-train the model solely on MCL. As shown in Table 4 we find that MTM plus MCL does improve from the baseline but is almost comparable to pre-training with only MCL. It shows the important role of TMCL in trajectory representation learning as it learns trajectory-map relationship and bridges the trajectory and map embedding space.

So our answer is "Yes, TMCL is crucial to improve trajectory representation."

Is MCL crucial for improving trajectory representation? Indeed, as shown in Table 4, when pre-training only with MCL, the improvement is 0.046. This makes sense in that HD-map is an important prior to prediction and thus better map representation in general can improve prediction. But is MCL also helpful to trajectory representation? To examine this, we only pre-train with TMCL. We find that without MCL, TMCL brings limited improvements compared with PreTraM, *e.g.,* 0.037 ADE-5 vs. 0.097 ADE-5 with PreTraM (Table. 4). This demonstrates that although map and trajectory are totally different modalities, PreTraM makes use of much more maps to enhance trajectory representation under the situation that the trajectory data is limited.

So our answer is "Yes, MCL is crucial to improve trajectory representation."

5 Related works

Given a trajectory forecasting model, the applicable pre-training schemes largely depend on the adopted scene representation. In this section, we first give a concise summary of the literature from the perspective of scene representation. Then, we review several works related to self-supervised learning for trajectory forecasting.

5.1 Scene Representation in Trajectory Forecasting

In complex urban traffic scenarios, it is crucial to utilize the semantic information of the scene to make accurate predictions. A widely-adopted approach is to employ rasterized top-down semantic images around the target agents as input and use CNNs to encode the context [5, 9, 27, 29, 35]. The past trajectories of the predicted agents are encoded separately and then aggregated with the context embedding. Our proposed PreTraM can be directly applied to pretrain models with this scene representation. The image-based representation has constant input size regardless of the complexity of the scene, which makes encoding simple and unified. However, some argued recently that rich semantic and structured information (e.g., relations between road segments) of the maps is lost through rasterization [13, 21]. To this end, they proposed to represent scenes as graphs that naturally inherit the structured information. Graph neural networks (GNN) [1, 21] and Transformers [13, 33] were then adopted to encode the context information from the scene graphs. Many graph-based models have then achieved state-of-the-art performance on multiple benchmarks [11, 15–17, 32, 36].

5.2 Self-Supervised Learning in Trajectory Forecasting

Pre-training and, in a broader sense, self-supervised learning are under-explored for trajectory forecasting. There are only a few recent works investigating their applications in trajectory forecasting. Inspired by similar methods in NLP, an auxiliary graph completion task was proposed in [13] to enhance the node representation, including both road elements and agents. However, the graph completion objective was jointly optimized with the prediction task. Moreover, the auxiliary task was applied to their Transformer-based encoder for the scene graphs, which limits the amount of data for self-supervised training to the size of prediction datasets. In contrast, our PreTraM framework lets trajectory encoder benefit from the large number of map patches that are not associated with agents. In [22], SimCLR was adopted to pre-train the representation of rasterized maps and agent relations. They deliberately introduced assumptions on semantic invariant operations based on domain knowledge. In our MCL, we follow [14] to avoid any assumptions on semantic invariant operations. Moreover, [22] focuses on contrastive learning within the same modality of data, and in the broader community of autonomous driving, [4] adopts single-modal contrastive learning on maps to improve sample efficiency, whereas our PreTraM framework leverages single-modal *and* cross-modal contrastive learning to jointly train trajectory and map representations. As shown in Sec. 4.5, PreTraM has clear performance advantage over single-modal contrastive learning within each data modality.

6 Discussion and Limitations

Our experiments demonstrated that PreTraM is effective for prediction models based on rasterized map representation. In principle, PreTraM is *not* limited to image-based map encoders. We can also apply PreTraM to those popular graph-based methods reviewed in Sec. 5.1, as long as we can obtain separate map and trajectory embeddings from the pipeline. For instance, some works adopted a two-stage graph encoding scheme, where the map graph was encoded before being fused with trajectory embeddings [15, 16, 21]. We want to point out that GNNs may behave differently from CNNs during pre-training, and we are interested in extending PreTraM to graph-based methods as future study. Meanwhile, other works integrate the road elements and agents into a single graph before aggregation [11, 13, 17, 32, 36]. PreTraM cannot be applied to these models, as there are no matching pairs of map and trajectory embeddings in their pipelines. We are interested in exploring alternative pre-training methods for these models. Besides, PreTraM may also benefit end-to-end methods that predict trajectories directly from raw sensor inputs [19, 30]. Cross-modal contrastive learning can still be applied to enhance trajectory representation with sensor inputs such as 3D point-clouds or images. Another interesting extension is to contrast multi-agent trajectory embeddings with maps to enhance interaction modeling [31].

7 Conclusion

In this paper, we propose PreTraM, a novel self-supervised pre-training scheme for trajectory forecasting. We design Trajectory-Map Contrastive Learning (TMCL) to help models capture the relationship between agents and the surrounding HD-map, and Map Contrastive Learning (MCL) to enhance map representation via a large number of augmented map patches that are not associated with the agents. With PreTraM, we reduce the error of Trajectron++ and AgentFormer by 5.5% and 6.9% relatively. Furthermore, PreTraM promotes data efficiency of the models. We also demonstrate that our method can consistently improve performance when the model size scales up. Through ablation studies and analysis, we show PreTraM indeed enhances map and trajectory representations. In particular, a better trajectory representation is learned via bridging the map and trajectory representations with TMCL, so that the trajectory encoder can benefit from the map representation enhanced by MCL. Therefore the performance improvement is attributed to the coherent integration of MCL and TMCL in our framework.

Acknowledgements

We sincerely appreciate Boris Ivanovic and Rowan McAllister for providing help on the experiments related to Trajectron++. This work was sponsored by Google-BAIR Commons program. Google also provided a generous donation of cloud compute credits through the Google-BAIR Commons program.

References

1. Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al.: Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261 (2018) 13
2. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. Advances in neural information processing systems **33**, 1877–1901 (2020) 2
3. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11621–11631 (2020) 2, 3, 4
4. Cai, P., Wang, S., Wang, H., Liu, M.: Carl-lead: Lidar-based end-to-end autonomous driving with contrastive deep reinforcement learning. arXiv preprint arXiv:2109.08473 (2021) 13
5. Chai, Y., Sapp, B., Bansal, M., Anguelov, D.: Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction. In: CoRL (2019) 9, 13
6. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: International conference on machine learning. pp. 1597–1607. PMLR (2020) 2, 4, 10
7. Chen, T., Kornblith, S., Swersky, K., Norouzi, M., Hinton, G.E.: Big self-supervised models are strong semi-supervised learners. Advances in neural information processing systems **33**, 22243–22255 (2020) 4
8. Chen, X., Fan, H., Girshick, R., He, K.: Improved baselines with momentum contrastive learning. arXiv preprint arXiv:2003.04297 (2020) 4
9. Cui, H., Radosavljevic, V., Chou, F.C., Lin, T.H., Nguyen, T., Huang, T.K., Schneider, J., Djuric, N.: Multimodal trajectory predictions for autonomous driving using deep convolutional networks. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 2090–2096. IEEE (2019) 9, 13
10. Deng, L.: The mnist database of handwritten digit images for machine learning research. IEEE Signal Processing Magazine **29**(6), 141–142 (2012) 2
11. Deo, N., Wolff, E., Beijbom, O.: Multimodal trajectory prediction conditioned on lane-graph traversals. In: Conference on Robot Learning. pp. 203–212. PMLR (2022) 13, 14
12. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). pp. 4171–4186. Association for Computational Linguistics, Minneapolis, Minnesota (Jun 2019). https://doi.org/10.18653/v1/N19-1423, https://aclanthology.org/N19-1423 2, 12
13. Gao, J., Sun, C., Zhao, H., Shen, Y., Anguelov, D., Li, C., Schmid, C.: Vectornet: Encoding hd maps and agent dynamics from vectorized representation. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020) 2, 13, 14
14. Gao, T., Yao, X., Chen, D.: SimCSE: Simple contrastive learning of sentence embeddings. In: Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing. pp. 6894–6910. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic (Nov 2021).

https://doi.org/10.18653/v1/2021.emnlp-main.552, https://aclanthology.org/2021.emnlp-main.552 4, 6, 13
15. Gilles, T., Sabatini, S., Tsishkou, D., Stanciulescu, B., Moutarde, F.: Home: Heatmap output for future motion estimation. In: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). pp. 500–507 (2021). https://doi.org/10.1109/ITSC48978.2021.9564944 13, 14
16. Gilles, T., Sabatini, S., Tsishkou, D., Stanciulescu, B., Moutarde, F.: THOMAS: Trajectory heatmap output with learned multi-agent sampling. In: International Conference on Learning Representations (2022), https://openreview.net/forum?id=QDdJhACYrlX 13, 14
17. Gu, J., Sun, C., Zhao, H.: Densetnt: End-to-end trajectory prediction from dense goal sets. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 15303–15312 (2021) 13, 14
18. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9729–9738 (2020) 2, 4, 10
19. Laddha, A.G., Gautam, S., Palombo, S., Pandey, S., Vallespi-Gonzalez, C.: Mvfusenet: Improving end-to-end object detection and motion forecasting through multi-view fusion of lidar data. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) pp. 2859–2868 (2021) 14
20. Li, J., Selvaraju, R.R., Gotmare, A.D., Joty, S., Xiong, C., Hoi, S.: Align before fuse: Vision and language representation learning with momentum distillation. In: NeurIPS (2021) 6
21. Liang, M., Yang, B., Hu, R., Chen, Y., Liao, R., Feng, S., Urtasun, R.: Learning lane graph representations for motion forecasting. In: European Conference on Computer Vision. pp. 541–556. Springer (2020) 13, 14
22. Ma, H., Sun, Y., Li, J., Tomizuka, M.: Multi-agent driving behavior prediction across different scenarios with self-supervised domain knowledge. In: 2021 IEEE International Intelligent Transportation Systems Conference (ITSC) (2021) 13
23. Ma, Y.J., Inala, J.P., Jayaraman, D., Bastani, O.: Likelihood-based diverse sampling for trajectory forecasting. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 13279–13288 (2021) 9
24. Ngiam, J., Vasudevan, V., Caine, B., Zhang, Z., Chiang, H.T.L., Ling, J., Roelofs, R., Bewley, A., Liu, C., Venugopal, A., Weiss, D.J., Sapp, B., Chen, Z., Shlens, J.: Scene transformer: A unified architecture for predicting future trajectories of multiple agents. In: International Conference on Learning Representations (2022), https://openreview.net/forum?id=Wm3EA5OlHsG 2
25. Van den Oord, A., Li, Y., Vinyals, O.: Representation learning with contrastive predictive coding. arXiv e-prints pp. arXiv–1807 (2018) 4
26. Phan-Minh, T., Grigore, E.C., Boulton, F.A., Beijbom, O., Wolff, E.M.: Covernet: Multimodal behavior prediction using trajectory sets. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 14074–14083 (2020) 9
27. Phan-Minh, T., Grigore, E.C., Boulton, F.A., Beijbom, O., Wolff, E.M.: Covernet: Multimodal behavior prediction using trajectory sets. In: 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020) 13
28. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International Conference on Machine Learning. pp. 8748–8763. PMLR (2021) 2, 3, 4, 6, 10

29. Salzmann, T., Ivanovic, B., Chakravarty, P., Pavone, M.: Trajectron++: Dynamically-feasible trajectory forecasting with heterogeneous data. In: European Conference on Computer Vision. pp. 683–700. Springer (2020) 3, 6, 7, 8, 9, 13
30. Shah, M., ling Huang, Z., Laddha, A.G., Langford, M., Barber, B., Zhang, S., Vallespi-Gonzalez, C., Urtasun, R.: Liranet: End-to-end trajectory prediction using spatio-temporal radar fusion. In: CoRL (2020) 14
31. Tang, C., Zhan, W., Tomizuka, M.: Exploring social posterior collapse in variational autoencoder for interaction modeling. Advances in Neural Information Processing Systems **34**, 8481–8494 (2021) 14
32. Varadarajan, B., Hefny, A., Srivastava, A., Refaat, K.S., Nayakanti, N., Cornman, A., Chen, K., Douillard, B., Lam, C.P., Anguelov, D., Sapp, B.: Multipath++: Efficient information fusion and trajectory aggregation for behavior prediction. In: 2022 International Conference on Robotics and Automation (ICRA). pp. 7814–7821 (2022). https://doi.org/10.1109/ICRA46639.2022.9812107 13, 14
33. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017) 13
34. Yuan, Y., Kitani, K.: Dlow: Diversifying latent flows for diverse human motion prediction. In: European Conference on Computer Vision. pp. 346–364. Springer (2020) 8, 9
35. Yuan, Y., Weng, X., Ou, Y., Kitani, K.: Agentformer: Agent-aware transformers for socio-temporal multi-agent forecasting. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2021) 3, 6, 7, 8, 9, 13
36. Zhao, H., Gao, J., Lan, T., Sun, C., Sapp, B., Varadarajan, B., Shen, Y., Shen, Y., Chai, Y., Schmid, C., Li, C., Anguelov, D.: Tnt: Target-driven trajectory prediction. In: Kober, J., Ramos, F., Tomlin, C. (eds.) Proceedings of the 2020 Conference on Robot Learning. Proceedings of Machine Learning Research, vol. 155, pp. 895–904. PMLR (16–18 Nov 2021), https://proceedings.mlr.press/v155/zhao21b.html 13, 14