TMLKD: Few-shot Trajectory Metric Learning via Knowledge Distillation

Danling Lai
School of Computer Science and Technology, Soochow University
Suzhou, China
dllai@stu.suda.edu.cn

Jiajie Xu*
Key Laboratory of Data Intelligence and Advanced Computing, Soochow University
Suzhou, China
xujj@suda.edu.cn

Jianfeng Qu
Key Laboratory of Data Intelligence and Advanced Computing, Soochow University
Suzhou, China
jfqu@suda.edu.cn

Pingfu Chao
Key Laboratory of Data Intelligence and Advanced Computing, Soochow University
Suzhou, China
pfchao@suda.edu.cn

Junhua Fang
Key Laboratory of Data Intelligence and Advanced Computing, Soochow University
Suzhou, China
jhfang@suda.edu.cn

Chengfei Liu
Swinburne University of Technology
Melbourn, Australia
cliu@swin.edu.au

ABSTRACT

Trajectory metric learning, which supports the trajectory similarity search, is one of the most fundamental tasks in spatial-temporal data analysis. However, existing trajectory metric learning methods rely on massive labels of pairwise trajectory distance, and thus cannot be applied to few-shot scenarios frequently occurring in real-world applications. Though performance drops caused by insufficient labels can be alleviated by knowledge distillation, we demonstrate that they cannot be directly applied to few-shot trajectory metric learning due to the domain shift problem. To this end, this paper proposes invariant and relaxed learning enhanced knowledge distillation method TMLKD for few-shot trajectory metric learning, such that domain-invariant representation and rank knowledge can be distilled. Specifically, in the representation learning phase, it first employs an adversarial sub-network to distinguish domain-specific and domain-invariant information, so as to distill transferable representation knowledge from teacher models. To mitigate the few-shot problem in student model training, we further enrich sparse labels of the target domain by utilizing the rank knowledge revealed in teachers' predictions. Particularly, TMLKD employs a list-wise learning-to-rank approach to learn the relaxed trajectory ranking orders instead of focusing on all the samples inefficiently. Finally, to guide accurate distillation, we adaptively assign reliability of teacher prediction by utilizing the ground-truth labels, to avoid misleading the student model with low-quality teacher predictions. Extensive experiments on three real-world datasets demonstrate the superiority of our model.

PVLDB Reference Format:
Danling Lai, Jiajie Xu, Jianfeng Qu, Pingfu Chao, Junhua Fang, and Chengfei Liu. TMLKD: Few-shot Trajectory Metric Learning via Knowledge Distillation. PVLDB, 18(8): 2308 - 2320, 2025.
doi:10.14778/3742728.3742729

*Corresponding author.

Proceedings of the VLDB Endowment, Vol. 18, No. 8 ISSN 2150-8097.
doi:10.14778/3742728.3742729

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/LynnTakesAHint/TMLKD.

1 INTRODUCTION

Fueled by the progress in GPS devices, a large amount of data, known as trajectories, are generated and benefit a wide range of real-life application domains, such as urban computing and behavior study. As the fundamental trajectory analytic problem, trajectory metric learning is crucial for various spatial-temporal data mining tasks, including trajectory similarity search, trajectory clustering [13] and trajectory prediction [17]. It aims to predict a similarity ranking based on the calculated trajectory distance. Classical methods compute trajectory similarity based on point-oriented matching with predefined measures, such as Dynamic Time Warping (DTW) distance [3], Fréchet distance [2], and Hausdorff distance [1]. However, these measures consume large space and suffer from high time complexity. For efficiency concerns, recent studies [40, 42–44] use learning-based approaches to learn measures with corresponding labels (e.g. pair-wise distance, ranking orders) by mapping trajectories into low dimensional space with deep learning models. Owing to the availability of sufficient labels, these methods significantly accelerate the trajectory similarity computation and achieve high accuracy.

Every sword has two edges. This reliance underscores the crucial need for meticulous and accurate data annotation, as the quality of the labels directly impacts the performance of these methods. However, it is challenging to obtain sufficient labels in real-world scenarios due to the high cost of manual annotation. Considering the data sparsity on the target measure, using predefined measures to enrich labels is also a method worth trying. However, this approach may suffer from weak generalization issues, because the focus of different measures differs, and the measure that accommodates a certain scenario may not be suitable for another. For instance, when monitoring airplane routes, we prioritize the **overall shape** of the trajectory to ensure the aircraft follows the planned route. In contrast, when delivering packages, our emphasis shifts to **local distances** to identify more efficient delivery methods. Despite the different emphases of measures in various scenarios, they still contain some insightful information. Data-sparse scenarios thus call for a few-shot trajectory metric learning strategy that can not only quickly learn from insufficient target measure labels but also make meaningful use of data-rich trajectory similarity measures in other domains.

Recently, knowledge distillation (KD) is known as one of the most successful approaches for few-shot learning [12, 28], with a basic idea to optimize a model (i.e., student model) by distilling the knowledge from well-trained models (i.e., teacher model). Two heads are better than one. Existing KD methods often adopt multiple teacher models, as they can correct the prediction error provided by a single teacher, thus achieving better performance [37, 41]. Owing to its generalization ability, KD has been successfully applied in various domains such as computer vision [35] and natural language processing [34]. Consequently, it provides great opportunities for trajectory similarity computation as well. Models on predefined source measures can be regarded as teacher models. These models come to our aid when training a more generalized student model with inadequate labels on the target measure. Despite this, directly applying multi-teacher distillation to trajectory metric learning tasks still faces some challenges:

- **How to obtain domain-invariant representations from teachers for enhanced representation learning in the target domain?** Existing methods [25, 30] in KD utilize the output of the last layer or that of the intermediate layers as representation knowledge to supervise the training of the student model. However, when applying this strategy to metric learning, the discrepancy across different measures necessitates for a selective distillation method to transfer representation knowledge from richly labeled source domains to the sparsely labeled target domain. Otherwise, directly transferring the representation knowledge may lead to sub-optimal results. Consequently, finding domain-invariant information is essential to integrate transferable knowledge from teacher models while bridging domain discrepancy. It thus calls for invariant learning-based feature extraction to discriminate shareable latent information for rational representation knowledge transfer [5, 7].
- **How to enrich sparse labels in the target domain by incorporating teacher predictions as soft labels selectively, such that knowledge from inconsistent teachers can be eliminated in sample-wise granularity?** Besides hard labels (i.e., annotated ranking order), the predictions of teacher models contain rich information, such as the ranking order of unlabeled trajectories. The inspiring knowledge can be exploited as soft labels. However, directly fusing this knowledge together is unreasonable, since different teacher models emphasize different aspects of the data and offer diverse results [19, 32]. Not all the teacher models provide equally valuable insights for every sample, so learning from various teacher models uniformly may result in sub-optimal results [41]. Consequently, it is essential to distinguish between high-quality and low-quality teachers at a sample-wise granularity, giving more weight to those teachers that closely align with the target domain for each specific sample. To effectively transfer ranking information from the teacher models, an accurate sample-wise confidence assessment of the teacher models is needed, enabling differentiation of their quality and supporting adaptive knowledge distillation.
- **How to employ the relaxed ranking matching strategy to distinguish between different labels and the varying importance of different trajectories, thereby ensuring performance and reducing noise?** The difficulty in data annotation hinders the training process of most existing trajectory similarity learners [42, 44], which requires informative soft labels from teacher models. When fully utilizing both hard labels and soft labels, their importance should be differentiated, since hard labels originate from precise annotations in the target domain, while soft labels are derived from the data transfer from other source domains. However, traditional methods fail to distinguish between these labels. Additionally, learning the detailed ranking of all unlabeled trajectories is both time-consuming and prone to introducing noise into the training process. To address this, we should prioritize the ranking of more similar trajectories while selectively ignoring the ranking of dissimilar ones, as they are less critical. Therefore, adopting a relaxed ranking matching approach would be advantageous, as it allows us to concentrate on the more relevant trajectories rather than inefficiently fusing all the samples with the ground truth.

To address the above challenges, we propose a new framework to cope with **T**rajectory **M**etric Learning with **K**nowledge **D**istillation, called **TMLKD**. **TMLKD** is general and principled to be naturally compatible with different architectures. Specifically, **TMLKD** uses multiple teacher models to transfer dual-channel knowledge from source domains to the student model on the target domain. To transfer shareable representation knowledge, we design adversarial sub-networks to capture domain-invariant knowledge, avoiding misleading the student with straightforward imitation tasks. Additionally, instead of directly fusing each teacher's predicted ranking results, we evaluate each teacher model's sample-wise confidence and use their predictions to enrich sparse labels, which serve as soft labels for rank knowledge transfer to the student model. With these enriched labels, we model correlations among trajectories and perform relaxed ranking distillation, emphasizing closely related trajectories. Our contributions can be summarized as follows:

- We design a general method for few-shot trajectory metric learning, which transfers knowledge from data-rich measures to the data-sparse measure. To the best of our knowledge, **TMLKD** is the first to introduce knowledge distillation for few-shot trajectory metric learning.
- We disentangle domain-invariant representation and domain-specific representation with the adversarial sub-networks during the teacher model training, to capture shareable representation knowledge. A contrastive constraint is further

utilized to prevent the shared and private latent features from interfering with each other.

- We perform a sample-wise confidence evaluation on each teacher, and enrich the sparse labels based on the confidence-aware predictions. Relaxed Ranking Distillation is further utilized to distill rank knowledge from closely related trajectories selectively.
- We conduct extensive experiments on three real-world trajectory datasets in order to demonstrate the effectiveness of our approach.

2 RELATED WORK

In this section, we outline recent research work related to TM-LKD from two aspects, including trajectory metric learning and knowledge distillation.

2.1 Trajectory Metric Learning

Trajectory metric learning is a fundamental step in trajectory data mining, in order to measure the similarity between a pair of trajectories. Traditional heuristic methods [1, 2, 9, 29, 32] aim to find optimal point matches. These methods count on hand-crafted rules and cannot utilize the information in trajectories. What is more, it usually has a large space and time complexity. There have been various techniques to accelerate the computation. Most techniques focus on pruning [16, 31] and designing approximate algorithms [11, 38].

As deep learning advances, trajectory representation learning [6, 14, 23] is developed to extract the spatial, structural, and time information of trajectories. Inspired by this, metric learning models are developed to transform trajectories into a d-dimensional embedding representation, simplifying the time complexity of similarity computation to $O(d)$. For example, NeuTraj [42] utilizes a spatial-memory-based LSTM unit to capture the similarity relationship between trajectories. Traj2SimVec [44] employs the distance information among sub-trajectories and acquires the most efficient matching correlation between points through a point-matching function. T3S [40] applies a multi-head self-attention network to capture the structural information of the trajectories, while an LSTM-based unit is exploited to extract spatial information. Traj-GAT [43] models each trajectory as a graph and captures long-term dependency based on graph attention modules.

Although these methods perform well on data-rich measures, they are incapable of modeling trajectories when labels for the target domain are insufficient, which leads to severe performance drop. It thus calls for an effective method to cope with few-shot metric learning via knowledge distillation.

2.2 Knowledge Distillation

Knowledge Distillation (KD) [18] is one of the most successful techniques in few-shot learning. The main idea of KD is to first train a high-performance but expensive teacher model, and then train a shallow student model to imitate the teacher model. The student model is optimized to learn from hard labels and soft labels. The hard labels are the ground-truth results, and the soft labels are the predicted results made by the teacher model. Despite the predicted results of teacher models, the representations produced by teacher models can also provide constructive knowledge for the student model [21, 22, 30].

In spatial-temporal data analysis, [10] proposes a KD model to solve trajectory-user linking problem, [26] transfers knowledge to forecast future path according to historic trajectory data, and [36] distills the knowledge to a mapless student network from a map-based prediction teacher network with a two-fold knowledge distillation framework, in order to predict the future coordinates of the given trajectory. Our technique differs from theirs, as we use KD to perform accurate trajectory metric learning when labels on the target measure are insufficient.

KD has achieved remarkable improvements when training the student model, especially with multiple teacher models included. Using multiple-teacher models helps reduce errors attributed to a single-teacher model, and enables the acquisition of more comprehensive knowledge. Several multi-teacher KD methods are proposed [15, 37, 41], they vary from each other in the way that they fuse different teacher models. Despite multi-teacher distillation can act as an effective method for few-shot metric learning, it lacks the consideration of domain shifts when transferring knowledge across different measures. We thus aim to improve existing methods by effectively distinguishing domain-invariant transferable representation knowledge and shareable rank knowledge from multiple teacher models.

3 PRELIMINARY

In this section, before delving into the details of our method, we first introduce the basic notations. Then we describe the related concepts and definitions. Finally, we describe the problem statement. Table 1 summarizes the frequent notations in the paper.

3.1 Definitions

Definition 3.1 (Trajectory). A trajectory T is a trace generated by a moving object. Usually, a trajectory is described by a sequence of points ordered by timestamps, i.e. $T = [p_1, p_2, ..., p_l]$, where p_i is the i-th location of the object.

Definition 3.2 (Trajectory Similarity Measure). A trajectory similarity measure $M(\cdot, \cdot)$ quantifies the similarity between two trajectories by calculating their distance. A smaller distance indicates a greater degree of similarity, thus influencing the relative ranking order in trajectory metric learning.

Definition 3.3 (Trajectory Metric Learner). Given labels under trajectory similarity measure M, a trajectory metric learner $f_M(\cdot, \cdot)$ aims to derive generic and informative representations of trajectories, to guide accurate trajectory metric learning.

It is important to emphasize that labels for trajectory metric learners can be divided into two categories: list-wise rank labels (e.g. ranking order of Top K trajectories) and distance labels (e.g. pair-wise distance for a pair of trajectories). Since the former fails to quantify the degree of similarity, using the latter can significantly improve the model performance. However, it is often time-consuming to calculate the pair-wise distance for a large-scale trajectory dataset.

Definition 3.4 (Source Measures and Target Measure). In our problem, we refer to a data-sparse measure in the real-world application

Table 1: Descriptions of notations used in this paper.

Notations	Description
$\mathcal{T}$	A database with trajectories and labels.
T	A trajectory in $\mathcal{T}$ consisting of a coordinate sequence.
M, f_M	A similarity measure for trajectory pairs and its metric learner.
$d(\cdot,\cdot)$	Euclidean distance between a pair of trajectory embeddings.
$\mathcal{G}$	Base model, such as T3S and NeuTraj.
$\mathcal{G}_j(T)$	Derived trajectory embedding via the j-th base model.
s, p	Domain-invariant trajectory representation.
p_j, h_j	Domain-specific trajectory representation and final trajectory representation of the j-th source domain.
W_s, W_p	Projection matrices to project s and p.
β	Learnable weight for domain-invariant and specific representations.
W_Q, W_K, W_v	The query, key and value matrices in self-attention mechanism for domain-invariant representation.
$\alpha_s, \alpha_p, \alpha_c$	Weights for teacher model losses.
$\hat{r}_t, \pi_t^g$	T-th trajectory's relevance score and ground-truth rank.
$\mathcal{N}_j^i, \mathcal{R}_j^i, c_j^i$	J-th teacher's nDCG, recall, and confidence scores for i-th sample.
rp_i, ρ	Rank-based similarity of i-th trajectory and top-position emphasis.
ξ	Threshold for partitioning unlabeled trajectories.
$r^a_{\pi_k}$	Predicted distance between k-th trajectory and anchor T_a.
$\pi^a_{K_1:K_2}$	Trajectories ranked between K_1 and K_2 for query T_a under the specific trajectory measure.

as the target measure. The labels of the target measure are usually limited and hard to access. Besides, we have several measures with sufficient labels, which are denoted as source measures. They usually have explicit formulas, which make the labels easy to calculate. Besides, they exhibit good generalization and indicate the spatial and structural information of trajectories. However, source measures may not perfectly fit the target domain and cannot be directly used in some scenarios.

3.2 Problem Statement

Given a trajectory database $\mathcal{T}$ and a target measure M_t with insufficient labels, we aim at deriving a trajectory distance learner f_t with the aid of several trajectory metric learner $f_{M_1}, f_{M_2}, \ldots$ on source measures $M_1, M_2, \ldots$, which are mainstream trajectory similarity measure and have easily computable labels. f_t should not only accommodate the limited annotated labels (i.e., the ranking orders) specific to the target domain but also effectively capture the correlation among data-sparse distance measure M_t and other data-rich source measures.

4 METHODOLOGY

4.1 Overview of TMLKD

Based on the learning-based trajectory metric learning framework, we extend it by incorporating knowledge distillation. Besides the base loss which is determined by the framework of the learner, we introduce latent knowledge and rank knowledge from teachers as guidance for distillation. The overall framework of our proposed TMLKD is shown in Figure 1. First, we introduce a trajectory similarity learner as our base model. We select different source measures with rich labels as multiple teacher models. During the training phase of teacher models, a discriminator-based network is designed to derive domain-invariant knowledge by learning the representation of teachers.

After training teacher models, we derive domain-specific trajectory representation from source measures, which can provide extra spatial and structural information when training the student model on the data-sparse measure. With confidence evaluation for each teacher in a sample-wise granularity, TMLKD then utilizes a confidence-aware correlation modeling module to enrich soft labels with the predicted result of teachers, where relaxed ranking loss is used to learn the rank knowledge from soft labels.

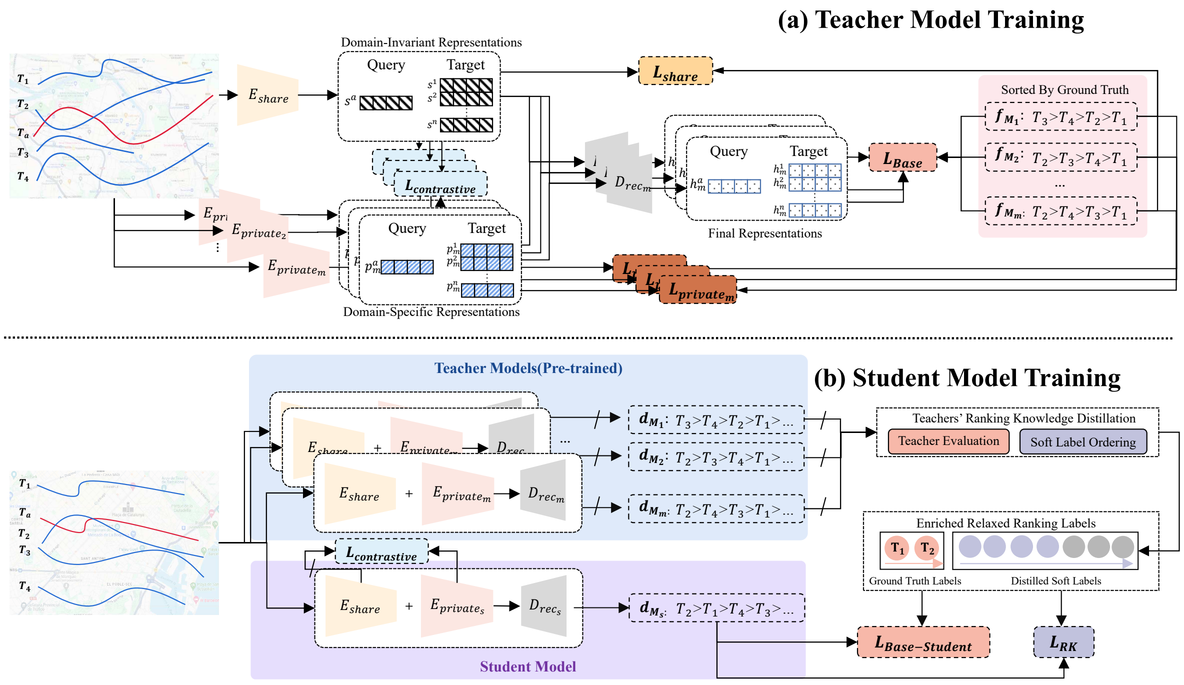
4.2 Case Study: Monitor Airplane Routes

To illustrate the proposed model, consider a real-world example with a large dataset of airplane routing trajectories. We have trained several teacher models on a set of airplane routes to learn various trajectory similarity metrics. TMLKD extracts domain-invariant representations, capturing shared trends across measures. Combined with domain-specific knowledge, these teacher models can quickly and accurately predict pairwise trajectory distances.

Now we aim to find routes with the same start and end points that closely match the shape of a query trajectory. However, the main challenge is that existing metrics are not well-suited to this specific scenario, and the difficulty of obtaining annotations causes the limited availability of labels. Despite having limited labels, Hausdorff [1] and Frechet [2] distances are useful for comparing overall shapes. We use the computed pairwise distances to generate soft labels for student model training. Additionally, pre-trained domain-invariant knowledge can be combined with domain-specific knowledge to transfer latent knowledge effectively.

4.3 Domain-invariant Representation Knowledge Extraction

Through the representation learning modules, teacher models are utilized to provide highly abstracted summaries of spatial and structural information within trajectories as latent knowledge. Traditional methods for distilling latent knowledge typically align the representations of teacher models with the student model after applying necessary projections [30], but they often overlook domain shifts when transferring knowledge from data-rich source domains to a data-sparse target domain. Even though teacher models are trained under different measures from different domains, they



(a) Teacher Model Training
Domain-Invariant Representations
Query
Target
E_{share}
L_{share}
Sorted By Ground Truth
f_{M_1}: T_3>T_4>T_2>T_1
f_{M_2}: T_2>T_3>T_4>T_1
f_{M_m}: T_2>T_4>T_3>T_1
$L_{contrastive}$
D_{rec_m}
L_{Base}
Final Representations
$E_{private_2}$
$E_{private_m}$
$L_{private_m}$
Domain-Specific Representations
Teacher Models(Pre-trained)
(b) Student Model Training
E_{share}
$E_{private_m}$
D_{rec_m}
d_{M_1}: T_3>T_4>T_2>T_1>...
d_{M_2}: T_2>T_3>T_4>T_1>...
d_{M_m}: T_2>T_4>T_3>T_1>...
Teachers' Ranking Knowledge Distillation
Teacher Evaluation
Soft Label Ordering
Enriched Relaxed Ranking Labels
Ground Truth Labels
Distilled Soft Labels
$E_{private_s}$
D_{rec_s}
d_{M_s}: T_2>T_1>T_4>T_3>...
$L_{Base-Student}$
L_{RK}
Student Model

Figure 1: Architecture of TMLKD. Part (a) demonstrates the training phase of teacher models, which aims at deriving transferable domain-invariant representation for the student model. Part (b) illustrates the training process of the student model, which utilizes the representation knowledge and rank knowledge from pre-trained teacher models.

can still provide constructive knowledge, as there is common and domain-invariant information to be extracted and utilized[7, 20]. To address this, we create two distinct feature spaces in each domain: one that captures common knowledge applicable across different domains, and another that records specific details relevant to tasks within each individual domain. Drawing inspiration from [24], we employ a "shared-private" architecture during training, consisting of two sub-networks: a *shared* sub-network E_{share} and a *private* sub-network $E_{private}$.

4.3.1 Shared Sub-network. The shared sub-network is utilized to extract domain-invariant knowledge. Its architecture follows the network structure of the base model. For instance, when T3S is used as the base model, the shared sub-network will consist of LSTM cells for extracting spatial information and attention modules for extracting structural information. Then, we employ an extra self-attention-based layer to model the domain-invariant representation. Specifically, we first derive the trajectory input $\hat{s} = \mathcal{G}(T)$ via base model $\mathcal{G}$, like T3S or NeuTraj. Then, we compute the domain-invariant representation s as follows:

$$(Q, K, V) = \hat{s} \times (W_Q, W_K, W_V) \tag{1}$$

$$A = \text{softmax}(\frac{Q \times K^{\text{T}}}{\sqrt{d}}) \tag{2}$$

$$s = A \times V \tag{3}$$

where Q, K, V are the query, key and value matrices, W_Q, W_K, W_V represents the weight matrices to produce Q, K, V. We further compute the attention score matrix A to derive the domain-invariant representation s.

When learning measures in different domains, the trajectory similarity learner uses the same shared sub-network for domain-invariant knowledge extraction. Hence, it should yield relatively balanced results across various source measures. We further formulate the loss for optimizing the shared sub-network as follows:

$$L_{share} = \sum_{j=1}^{m} (d(s_a, s_b) - f_{M_j}(T_a, T_b))^2 \tag{4}$$

where f_{M_j} denotes the trajectory distance learner on the j-th source measure M_j.

4.3.2 Private Sub-network. For extracting domain-specific knowledge, we employ private sub-networks. Each trajectory similarity learner on each domain, learning a specific measure, has a unique private sub-network to learn domain-specific information. It maintains consistency with the network structure of the base model as well. With the j-th private sub-network, we compute the domain-specific representation p_j obtained with the private sub-network as follows:

$$p_j = \mathcal{G}_j(T), j \in \{1, 2, ..., m\} \tag{5}$$

where $\mathcal{G}_j$ refers to the base model training with the $j - th$ source measure. Since the private sub-network extracts domain-specific knowledge and is supposed to show the most optimal performance on the corresponding measure, we optimize the private sub-network with $L_{private}$:

$$L_{private} = \sum_{j=1}^{m} (d(p_{j_a}, p_{j_b}) - f_{M_j}(T_a, T_b))^2 \tag{6}$$

Finally, we use a learning-based combination D_{rec} to obtain the final trajectory representation:

$$h_j = \beta \cdot (W_s \cdot s) + (1 - \beta) \cdot (W_p \cdot p_j) \tag{7}$$

where W_s, W_p are the representation projection matrices. β is a learnable parameter to balance the weight of domain-specific knowledge and domain-invariant knowledge.

4.3.3 Incorporating Adversarial Strategy. While the shared-private model segregates the feature space into shared and private domains, there's no assurance that features eligible for sharing cannot exist within the private feature space, or vice versa. To avoid task-invariant features appearing in both shared space and private space, we introduce orthogonality constraints [4] to prevent the shared sub-network and private ones from storing redundant knowledge:

$$L_{contrastive} = \sum_{j=1}^{m} ||S^T P_j||_F^2 \tag{8}$$

where $||...||_f^2$ is the squared Frobenius norm, m is the size of source measures. The rows of S and P_j are the output of the shared sub-network and that of the j-th private sub-network respectively. By minimizing $L_{contrastive}$, we encourage the domain-invariant representation from shared sub-network and domain-specific knowledge from each private sub-network to encode different perspectives of trajectories, in order to achieve Knowledge Segregation. Eventually, the final loss of the teacher training phase can be written as follows:

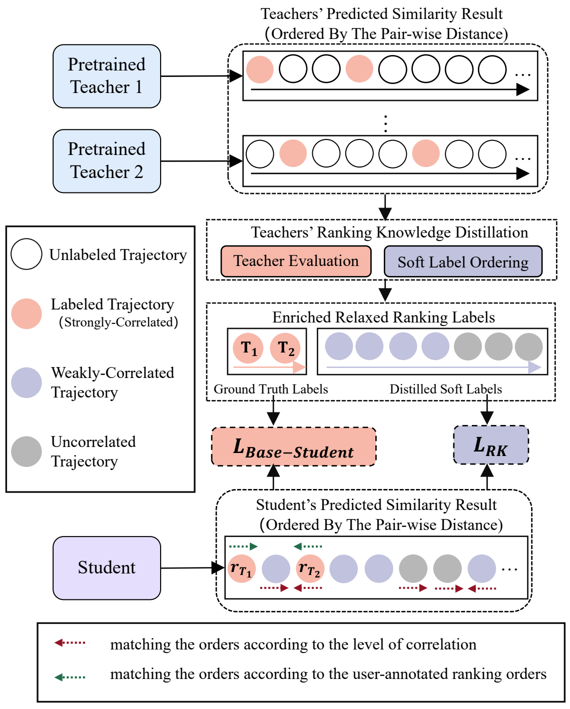
$$L = L_{Base} + \alpha_s L_{share} + \alpha_p L_{private} + \alpha_c L_{contrastive} \tag{9}$$

where L_{Base} refers to the loss of teacher model, it depends on the definition of teacher model. For example, L_{base} will be the pair-wise distance loss when using Neutraj [42] as base model, and triplet loss when employing Traj2Simvec [44]. $\alpha_s, \alpha_p, \alpha_c$ are hyper-parameters. After the teacher models' training phase, the shared-private sub-network architecture on source measures will be further utilized in the student model training phase to provide representation knowledge and rank knowledge for distillation. However, the parameters of sub-networks will not be updated when training student models, which means the domain-invariant representation is not updatable during student model training.

4.4 Shareable Rank Information Distillation

Given the label sparsity in the target domain, KD modules are needed to enhance the labels by incorporating the predictions of teacher models as soft labels. However, directly fusing the output of teacher models will lead to a performance drop because of the discrepancy among measures. When dealing with multiple teacher models from different source domains, it is essential to first identify the higher-quality teachers through accurate confidence evaluation. Then, we further utilize a list-wise learning-to-rank method to transfer relaxed rank knowledge to the student model.

4.4.1 Sample-Wise Confidence-Aware Teacher Selection. Knowledge distillation aims to seek useful information from teacher models as extra guidance for the student model. However, directly transferring knowledge may result in sub-optimal results. For one thing, there may be a significant difference in the similarity predictions between the student measure and the teacher measures. For another, the predictions from the teacher measures are not the same either. Setting equal weights for all the teacher measures may result in a performance drop. Therefore, it is crucial to accurately measure the teacher models' confidence and conduct biased knowledge distillation based on the results. During the evaluation, sparse labels should be fully utilized to derive precise teacher-student measure confidence evaluation. However, since labels are insufficient, aggregating confidence across multiple samples to form an overall evaluation result may be inaccurate, as it neglects the diversity of samples and can lead to over-fitting. It requires to adaptively assign weight at a sample-wise granularity. Hence, given anchor trajectory T_a, we employ teacher models to predict its top-k similar trajectories, where k refers to the number of trajectories with similarity ranking annotations for T_a. We employ nDCG to evaluate the quality of the predicted ranking order of each teacher model, with the relevance score $\hat{r}_t$ defined as follows:



Teachers' Predicted Similarity Result
(Ordered By The Pair-wise Distance)
Pretrained Teacher 1
Pretrained Teacher 2
Unlabeled Trajectory
Labeled Trajectory (Strongly-Correlated)
Weakly-Correlated Trajectory
Uncorrelated Trajectory
Teachers' Ranking Knowledge Distillation
Teacher Evaluation
Soft Label Ordering
Enriched Relaxed Ranking Labels
T_1
T_2
Ground Truth Labels
Distilled Soft Labels
$L_{Base-Student}$
L_{RK}
Student's Predicted Similarity Result
(Ordered By The Pair-wise Distance)
Student
r_{T_1}
r_{T_2}
matching the orders according to the level of correlation
matching the orders according to the user-annotated ranking orders

Figure 2: The Illustration of Soft Label Enrichment and Relaxed Ranking Loss.

$$\hat{r}_t = \begin{cases} \phi \cdot (k - t), & \pi_t^g \in 1, 2, ..., k \\ 0, & \pi_t^g \in k+1, k+2, ... \end{cases} \tag{10}$$

where π_t^g refers to the t-th trajectory's ranking order according to the ground truth. ϕ is a dataset-dependent hyper-parameter, which is set to 0.01 according to our experiments. Besides nDCG, we utilize the recall rate $\hat{recall}_t$ to evaluate the recovery ability of teacher models. Specifically, we compute the occurrences of the top-k trajectories from the ground truth in the predicted top-$\hat{k}$ trajectories by the teacher model, where $\hat{k}$ is usually larger than k to ensure a certain level of fault tolerance.

To balance the magnitude of nDCG and Recall Rate, we define $\mathcal{N}_j^i = \text{nDCG}_j^i / \sum_{k=1}^{m} \text{nDCG}_k^i$ as the final nDCG score of j-th teacher model on the i-th trajectory sample. Then we compute the Recall score $\mathcal{R}_j^i$ in a similar way. Eventually, we derive the confidence score c_j^i as follows:

$$c_j^i = \frac{\mathcal{N}_j^i \cdot \mathcal{R}_j^i}{\sum_{k=1}^m \mathcal{N}_k^i \cdot \mathcal{R}_k^i} \tag{11}$$

4.4.2 Confidence-aware Soft Label Enrichment. Given an anchor trajectory and the sparse manual annotation, the trajectories in the datasets can divided into two categories, namely labeled ones and unlabeled ones. In order to extract rich information in a large number of trajectories, we, inspired by [39], introduce enriched soft labels to learn the knowledge of ranking orders of unlabeled items while preserving the priority of annotated labels. Hence, they possess the most vital priority and are supposed to be positioned at the forefront. However, they suffer from data sparsity and contribute little to the training of the student model.

In contrast, even though trajectories in the latter categories are not most closely related to the target measure, they reveal additional information about the ranking of samples. Teacher models, demonstrating superior performance, can provide supplementary knowledge to enhance the distillation. Specifically, we utilize teacher models to make predictions as soft labels for a given anchor trajectory. The soft labels enrichment process includes two steps, i.e. *rank-based correlated score generator* and *correlation-based unlabeled trajectories sampling.*

Rank-based Similarity Correlated Score Generator First, we assume that unlabeled trajectories predicted by the teacher model to be at the top positions are more likely to be correlated with the query trajectory and are more likely to be similar to it. Hence, we develop the rank-based similarity score as follows:

$$rp_i = \frac{1}{\pi_i} \tag{12}$$

where π_i denotes the ranking order of target trajectory T_i. However, the decay rate of the above-mentioned method is too rapid. As π_i increases, rp_i becomes extremely small. This is not suitable for certain measures with low discriminative power. Therefore, we further define rp_i as follows:

$$rp_i = \sum e^{\frac{-\pi_i}{\rho}} \tag{13}$$

where ρ is a dataset-dependent hyper-parameter that adjusts the emphasis on top positions. For example, when ρ is small, the top positions will be paid more attention, while when ρ gets larger, it gradually turns into a uniform distribution.

Correlation-based Unlabeled Trajectories Sampling For the same target trajectory, its ranking position often varies in the predictions of different teacher models, leading to different similarity scores. In such cases, it is important to focus on the teacher model with higher confidence and lean towards its predictions. In specific, according to the probability of a trajectory T predicted by the i-th teacher model to be selected as the k-th most similar, we define the final correlation score as follows:

$$\text{score}_i = \sum_{j=1}^{m} c_j^i \cdot (e^{\frac{-k}{\rho}}) \tag{14}$$

The above probabilities can further differentiate unlabeled trajectories by combining teachers' evaluated confidence and the trajectory's rank position in the predictions. Then, we set a threshold ξ and further define: labeled trajectories are strongly correlated trajectories (SC), trajectories with a selection probability greater than ξ are weakly correlated trajectories (WC), and those with probabilities less than ξ are considered uncorrelated ones (U).

4.4.3 Relaxed Ranking Loss For Learning Soft Labels. When computing loss on training pairs for optimization, it is not only ineffective but also time-consuming to take all the target trajectories into consideration. Hence, most learning-based metric learning methods often focus on the most similar and dissimilar trajectory pairs [42] since they are more discriminative, or use random sampling methods [44] to reduce the pairs included during the computation. However, directly adopting traditional loss is impossible in our problem, as only relative ranking orders on target trajectories are given, which hinders the utilization of pair-wise distance information when computing loss of learning-based methods. What is more, limited by the quality of negative samples in few-shot learning, a traditional pair-wise loss like triplet loss may lead to sub-optimal results.

When lacking pair-wise distance labels, list-wise learning-to-rank tasks can be considered as another viable alternative. In concrete, based on the pair-wise distance between trajectory representations, these tasks represent the likelihood of a ranking order via a defined permutation probability and then train the model, in order to maximize the probability of matching the ground-truth ranking order. Inspired by [21], we adopt the relaxed ranking loss (RRD) to not only learn from limited ranking order information but also speed up the convergence by focusing on more discriminative trajectory pairs.

Figure 2 illustrates the process of enriching soft labels and performing relaxed ranking loss. The designed RRD module defines a relaxed permutation probability inspired by [21] in order to maintain the "SC-WC-U" trajectory order. Specifically, strongly correlated trajectories have clear labels and should be placed at the top positions according to the labels. As for the remaining trajectories, we only need to ensure that WC comes before U, without emphasizing their internal relationships. The relaxed permutation probabilities can be formulated as below:

$$r_{\pi_k}^a = d(h(T_a), h(T_k)) \tag{15}$$

$$p(\pi_{1:K_1}^a | r) = \prod_{k=1}^{K_1} \frac{\exp(-r_{\pi_k}^a)}{\sum_{i=k}^m \exp(-r_{\pi_i}^a)} \tag{16}$$

$$p(\pi_{K_1:K_2}^a | r) = \prod_{k=K_1}^{K_2} \frac{\exp(-r_{\pi_k}^a)}{\sum_{i=K_2}^m \exp(-r_{\pi_i}^a)} \tag{17}$$

where $d(,)$ denotes the Euclidean distance between the embeddings, and $\pi_{1:K_1}^a$ denotes the Top K similar trajectories for the anchor trajectory. Here, among the m target trajectories, the top K_1 trajectories are SC ones for the anchor trajectory, and trajectories ranked between K_1 and K_2 are WC ones. By maximizing the log-likelihood, we not only maintain the detailed ranking orders among labeled trajectories but also distill useful knowledge from enriched soft labels predicted by teacher models, which allows for some tolerance. During the training of the student model, the pre-trained teacher will not be updated, and the domain-invariant representation will remain unchanged as well.

5 EXPERIMENTS

In this section, we briefly present the datasets used in our experiments. Then we introduce experimental settings and evaluation metrics in our paper. Then, we give an introduction of the baseline methods and show the experimental results of all the models in order to compare our TMLKD with them. Additionally, we analyze the accuracy and efficiency of our method and study the influence of the components of our model. Finally, we discuss the impact of some hyper-parameters used in our model. We report the major results with the analysis for the following questions:

- **RQ1:** Does TMLKD mitigate the performance drop on target measure which is due to insufficient labels?
- **RQ2:** What is the performance of TMLKD compared with other multi-teacher knowledge distillation methods?
- **RQ3:** What are the impacts of our designed modules on the model performance? For instance, how do rank knowledge distillation, and representation knowledge transfer influence our model?
- **RQ4:** What are the advantages of TMLKD in terms of efficiency when performing trajectory similarity computation and similar trajectory search?
- **RQ5:** How do the hyper-parameters, such as embedding dimension d, affect the effectiveness of our model?

5.1 Experiment Setup

5.1.1 Datasets. Our experiments are based on three real-world public trajectory datasets:

- **Porto** [27] is a dataset that contains 1.7 million vehicle trajectories. It is generated from 442 taxis in Porto, Portugal from 2013 to 2014.
- **Rome**[1] is a dataset made up of 367,052 taxi trajectories in Rome, Italy. The trajectories cover over 30 days.
- **Chengdu** contains over 1.4 billion taxi GPS data points generated by more than 14,000 taxis in Chengdu, China, which is provided by DiDi GAIA.

Following [40, 42], we choose trajectories in the city center and partition the whole area into 1100×1100 sized grids. Then we remove the trajectories of less than 10 records.

5.1.2 Experimental Guidance. During training, we evaluate models trained with different source measures (DTW [3], ERP [8], Hausdorff [1], and Fréchet distance [2]) based on their prediction results on labeled data in the target domain and remove the source measure with the lowest hitting rate (HR@5). For training purposes, we focused on DTW, Hausdorff, Fréchet, and SSPD as our target measures. While training the model for a specific target measure, we select from the remaining provided measures as source measures. The trainable parameters are optimized by Adam based on the training set. When training the teacher model, the sample size is set to 30. Different base models require different batch sizes and learning rates, which we have explained in detail in the code.

When training the teacher models, we randomly select $15k$ trajectories as our dataset. Then, we partition the overall dataset into the training set, validation set and test set following a ratio $3:1:6$. What is more, all the pair-wise distance of the trajectories in the training set is available when training teacher models.

When training the student model, we only randomly select $10k$ trajectories as our dataset, ensuring that the selected data has a similar distribution to the training data of the teacher model. Given that users cannot annotate massive amounts of data in batches, we randomly select $3k$ trajectories as the training set, and the rest will be used as the test set. We simulate user annotations using the following steps. First, we randomly divide the training set into several groups, with each group containing 50 trajectories. Second, we select one trajectory within each group, then we search for the top 5 similar trajectories from the remaining unselected trajectories according to the target measure. The selected trajectory and the searched ones form a *similar cluster* In each group. Afterward, we select 4-5 non-overlapping similar clusters in each group. Finally, this process yields approximately 240-260 similarity clusters on the training set.

For users in real-world scenarios, it is hard to annotate the distance between trajectories, as they can only provide the relative ranking orders of the search results. Therefore, during the training stage of the student model, we only provide the ranking order between anchor (query) trajectory and target trajectories as labels, while the pair-wise distance is not available. Each experiment was run on a single NVIDIA P40 GPU (24GB) without sharing.

5.1.3 Evaluation Metrics. As in [40, 42–44], hitting rate and recall rate are used to evaluate the accuracy of similar trajectory search:

- **Hitting Rate@K** assesses the percentage of overlap between the predicted Top K results and the ground truth, indicating greater accuracy with a higher ratio.
- **Recall K@R** assesses the recovery of Top-K ground-truth trajectories by the Top-R results. Enhanced recall signifies an improved ability to recover trajectories.

We report HR@10, HR@50, R10@50 as our evaluation metrics. Higher values indicate better model performance.

5.2 Baselines

5.2.1 Teacher/Student Base Models. We select our base models from existing learning-based trajectory similarity computation methods:

- **NeuTraj** [42] introduces a spatial memory unit to LSTM modules in order to capture correlations between trajectories. A distance-weighted ranking loss is also employed to support accuracy and fast convergence.
- **Traj2SimVec** [44] utilizes a point-matching auxiliary task to leverage distance information between sub-trajectories and optimally learns point-to-point matching.
- **T3S** [40] employs vanilla multi-head self-attention network and LSTM modules to capture spatial and structural characteristics of trajectories.

[1]https://crawdad.org/roma/taxi/20140717/

Table 2: Overall Performance Comparison for Non-KD Methods and Our Method.

Dataset	Method	SSPD			Hausdorff			DTW			Discrete-Frechet		
		HR@10	HR@50	R10@50	HR@10	HR@50	R10@50	HR@10	HR@50	R10@50	HR@10	HR@50	R10@50
Porto	Traj2SimVec	0.2132	0.2475	0.4764	0.1920	0.2362	0.4527	0.3891	0.4925	0.76800	0.4147	0.5167	0.7786
	TrajCL	0.2071	0.2749	0.4799	0.1801	0.2517	0.4305	0.2400	0.3043	0.5216	0.2086	0.2702	0.4514
	NeuTraj	0.2011	0.2728	0.4755	0.2004	0.2720	0.4847	0.4152	0.4975	0.7681	0.4167	0.5350	0.7331
	NT-No-SAM	0.2530	0.2907	0.5365	0.2609	0.3111	0.5944	0.4392	0.5231	0.7899	0.4714	0.5677	0.7729
	NNS-TMLKD	0.3675	0.3919	0.7005	0.3221	0.3664	0.6943	0.5116	0.5720	0.8393	0.4442	0.5351	0.8035
	Improv.	45.24%	34.82%	30.58%	23.48%	17.79%	16.82%	16.48%	9.34%	6.25%	-5.78%	-5.75%	3.96%
	T3S	0.2609	0.3119	0.5642	0.2294	0.2917	0.5340	0.4411	0.5358	0.8165	0.4327	0.5297	0.7697
	T3S-TMLKD	0.4055	0.4638	0.7599	0.3382	0.4214	0.7201	0.5007	0.5661	0.8395	0.5032	0.5769	0.8180
	Improv.	55.47%	48.68%	34.70%	47.44%	44.45%	34.85%	13.51%	5.65%	2.82%	16.28%	8.91%	6.28%
Rome	Traj2SimVec	0.2314	0.2817	0.5331	0.1738	0.2271	0.4377	0.3250	0.4041	0.6831	0.2705	0.3531	0.5600
	TrajCL	0.3082	0.4014	0.6587	0.2493	0.3256	0.5300	0.3287	0.3772	0.6135	0.2374	0.2739	0.4527
	NeuTraj	0.1970	0.2891	0.4672	0.1727	0.2769	0.4464	0.3416	0.4764	0.7250	0.4059	0.5395	0.7587
	NT-No-SAM	0.2610	0.3194	0.5714	0.2544	0.3246	0.5922	0.4474	0.5466	0.7958	0.4390	0.5272	0.7632
	NNS-TMLKD	0.2774	0.3302	0.5850	0.2824	0.3476	0.6252	0.5106	0.5859	0.8437	0.5057	0.5946	0.8300
	Improv.	6.28%	3.38%	2.37%	10.99%	7.07%	5.59%	14.14%	7.19%	6.01%	15.21%	12.78%	8.75%
	T3S	0.2602	0.3308	0.5796	0.2342	0.3137	0.5643	0.4463	0.5478	0.8092	0.4041	0.5067	0.7304
	T3S-TMLKD	0.3596	0.4287	0.7090	0.2915	0.3865	0.6461	0.5057	0.5795	0.8389	0.4943	0.5735	0.7959
	Improv.	38.20%	29.57%	22.32%	24.46%	23.24%	14.50%	13.32%	5.78%	3.67%	22.32%	13.18%	8.96%
Chengdu	Traj2SimVec	0.1161	0.1778	0.3223	0.1330	0.2103	0.3873	0.1427	0.2034	0.3403	0.2836	0.3838	0.6071
	TrajCL	0.1574	0.2542	0.4196	0.1322	0.2125	0.3404	0.1596	0.2293	0.3832	0.1106	0.1631	0.2583
	NeuTraj	0.1244	0.2188	0.3458	0.1349	0.2433	0.3794	0.2635	0.3785	0.6047	0.2780	0.4205	0.6143
	NT-No-SAM	0.1559	0.2419	0.3947	0.1799	0.2856	0.4703	0.2814	0.3862	0.6206	0.3846	0.5017	0.6654
	NNS-TMLKD	0.1996	0.2784	0.4851	0.2252	0.3140	0.5610	0.3503	0.4400	0.6903	0.4319	0.5295	0.7993
	Improv.	28.02%	15.12%	22.91%	25.18%	9.95%	19.30%	24.50%	13.95%	11.23%	12.28%	5.55%	20.13%
	T3S	0.2126	0.3264	0.5368	0.2388	0.3634	0.5763	0.3051	0.4153	0.6527	0.3358	0.4294	0.6865
	T3S-TMLKD	0.2602	0.3509	0.5942	0.2488	0.3634	0.6049	0.4136	0.4757	0.7299	0.4100	0.4982	0.7193
	Improv.	22.35%	7.51%	10.69%	4.19%	0%	4.95%	35.57%	14.54%	11.83%	22.08%	16.04%	4.78%

- **TrajCL** [6] uses a dual-feature self-attention-based trajectory encoder. We follow the paper to fine-tune the encoder with a two-layer MLP.

Besides, we implement the variants of NeuTraj: **NT-No-SAM**, which only removes the SAM module of NeuTraj. The embedding procedure of LSTM and weighted sampling strategies remain utilized during the training. We replace the LSTM cell with GRU cell.

The source code of NeuTraj and TrajCL is available online, so we directly use it. For other methods, we follow the settings in the paper and implement these methods on our own. Based on the performance of the base model under data sparsity, we select the strongest-performing T3S and NT-No-SAM as base models to further investigate the effectiveness of the TMLKD framework.

5.2.2 *KD Strategy.*

- **Single-Teacher** [18, 30] is referred to as Vanilla KD. The student only learns from one teacher [33]. Here, we choose the teacher model with the highest score as our teacher.
- **Random-Teacher Ensemble** [15] randomly selects one teacher from several teachers in every mini-batch to provide guidance. It is different from our method as we select the teacher based on the confidence score.
- **Equal-Weight Ensemble** [41] assigns equal weight to each teacher and learns from the average output of teachers.
- **Fixed-Weight Ensemble** [37] enables the student to learn from the aggregation result of teacher models. The weights are fixed during training based on the priority calculated by our confidence evaluation module.

5.3 Performance Comparison (RQ1-RQ3)

5.3.1 Accuracy Evaluation (RQ1-RQ2). In this section, we validate the superiority of TMLKD on three datasets. Table 2 demonstrates the results of our method, where NNS-TMLKD and T3S-TMLKD mean TMLKD method with NT-No-SAM and T3S as the base model, respectively. According to the result, we note the following key observations. First, existing methods require pair-wise distance labels for training. Hence, insufficient rank labels on the target measure lead to a severe performance drop on all the datasets. For example, for the dataset *Chengdu*, when using NeuTraj under SSPD measure, the recall rate drops to 0.3458, which is unsatisfactory.

Second, it can be concluded from the table that TMLKD mitigates the performance drop to a certain extent. We further demonstrates in the table the improvements of TMLKD compared to the best-performing non-KD model (see in *Improv.*). Such huge improvements demonstrate the effectiveness of our model, which is enhanced by transferring domain-invariant representation knowledge and rank knowledge from source measures with rich labels.

Furthermore, we compare TMLKD with other knowledge distillation strategies in Table 3. Although these methods all introduce knowledge to the student model, their performance varies. Single Teacher introduces knowledge from a single teacher model, but it is not comprehensive enough compared to multi-teacher approaches. The fixed-weight strategy performs the worst because it overlooks sample bias in small-sample scenarios, leading to overfitting and subsequent performance degradation. When utilizing multiple-teacher models, the random-weight strategy enhances the performance of the student model by incorporating knowledge from multiple teachers. However, since it arbitrarily selects teacher models without considering teacher-student consistency, it still suffers from performance loss. Similarly, the average-weight strategy fuses multiple teacher models without distinguishing high-quality teachers, resulting in only a slight performance improvement. Given the unique challenges of few-shot learning, our method does not achieve the best performance in all cases, but overall it demonstrates a significant advantage. By dynamically adjusting the weights of teacher models and thoroughly considering sample-level confidence

Table 3: Performance Comparison with Different Teacher Ensemble Strategies, with T3S as the Base Model.

Dataset	Method	SSPD			Hausdorff			DTW			Discrete-Frechet			Average
		HR@10	HR@50	R10@50	HR@10	HR@50	R10@50	HR@10	HR@50	R10@50	HR@10	HR@50	R10@50	Rank
Porto	Single	0.4012	0.4409	0.7449	0.3377	0.4172	0.7143	0.4725	0.5232	0.8073	0.5102	**0.5802**	0.8196	3
	Random	0.4028	0.4572	0.7557	0.3263	0.3911	0.7010	**0.5058**	**0.5697**	**0.8432**	0.4928	0.5625	0.8068	4
	Average	**0.4087**	**0.4664**	**0.7607**	0.3327	0.4122	0.7108	0.4493	0.4908	0.7715	0.5072	**0.5827**	**0.8198**	2
	Fixed	0.4011	0.4529	0.7528	0.3286	0.3976	0.7033	0.4865	0.5537	0.8291	0.4941	0.5660	0.8148	5
	TMLKD	0.4055	0.4638	0.7599	**0.3382**	**0.4214**	**0.7201**	0.5007	0.5661	0.8395	0.5032	0.5769	0.8180	**1**
Rome	Single	**0.3644**	**0.4334**	**0.7174**	0.2886	0.3819	0.6364	**0.5063**	0.5790	0.8361	0.4831	0.5563	0.7750	3
	Random	0.3560	0.4224	0.7055	0.2906	0.3826	0.6439	0.4896	0.5679	0.83000	0.4852	0.5637	0.7885	4
	Average	0.3556	0.4234	0.7036	0.2904	**0.3875**	0.6388	0.4989	0.5754	0.8357	**0.4965**	**0.5790**	**0.7994**	2
	Fixed	0.3529	0.4227	0.7043	0.2826	0.3705	0.6288	0.5037	0.5783	0.8387	0.4902	0.5711	0.7946	4
	TMLKD	0.3596	0.4287	0.7090	**0.2915**	0.3865	**0.6461**	0.5057	**0.5795**	**0.8389**	0.4943	0.5735	0.7959	**1**
Chengdu	Single	0.2429	0.3230	0.5623	0.2454	0.35500	0.5975	0.4084	0.4677	0.7235	0.4119	0.4962	0.7164	3
	Random	**0.2612**	0.3493	0.5928	0.2374	0.3402	0.5891	0.3330	0.4051	0.6611	0.4087	0.4945	0.7200	4
	Average	0.2485	0.3323	0.5715	0.2445	0.3558	0.6002	0.3328	0.4074	0.6628	0.4119	0.4999	**0.7201**	2
	Fixed	0.2423	0.3212	0.5623	0.2381	0.3444	0.5888	0.3336	0.4084	0.6624	**0.4126**	**0.5003**	0.7177	5
	TMLKD	0.2602	**0.3509**	**0.5942**	**0.2488**	**0.3634**	**0.6049**	**0.4136**	**0.4757**	**0.7299**	0.4100	0.4982	0.7193	**1**

Table 4: Results for Ablation Experiments for Different Methods, with T3S as the Base Model.

Dataset	Method	SSPD			Hausdorff			DTW			Discrete-Frechet		
		HR@10	HR@50	R10@50	HR@10	HR@50	R10@50	HR@10	HR@50	R10@50	HR@10	HR@50	R10@50
Porto	TMLKD w/o RK	0.3967	0.4267	0.7349	0.3250	0.3845	0.7022	0.4789	0.5138	0.7916	0.5053	0.5053	0.8184
	TMLKD w/o *s*	0.2473	0.2923	0.5371	0.2306	0.2878	0.5387	0.4544	0.5488	0.8329	0.4268	0.5251	0.7585
	TMLKD w/o Lcon	0.3988	0.4626	0.7519	**0.3428**	0.3874	**0.7221**	0.4612	0.5306	0.8078	**0.5099**	**0.5804**	**0.8245**
	TMLKD w/o Attn	0.3719	0.4087	0.7238	0.3340	0.4001	0.7103	0.4567	0.5337	0.8101	0.4937	0.5654	0.8117
	TMLKD	**0.4055**	**0.4638**	**0.7599**	0.3382	**0.4214**	0.7201	**0.5007**	**0.5661**	**0.8395**	0.5032	0.5769	0.8180
Rome	TMLKD w/o RK	0.3569	0.4250	0.7044	0.2899	0.3750	0.6412	0.4773	0.5563	0.8210	**0.4978**	**0.5796**	0.7882
	TMLKD w/o *s*	0.2621	0.3342	0.5848	0.2379	0.3249	0.5624	0.4341	0.5382	0.8012	0.4240	0.5315	0.7640
	TMLKD w/o Lcon	0.3479	0.4168	0.6933	**0.3432**	**0.4430**	**0.6980**	0.4933	0.5463	0.7987	0.4224	0.4999	0.7468
	TMLKD w/o Attn	0.3225	0.3784	0.6570	0.3198	0.4133	0.6931	0.4911	0.5541	0.8039	0.4348	0.5259	0.7677
	TMLKD	**0.3596**	**0.4287**	**0.7090**	0.2915	0.3865	0.6461	**0.5057**	**0.5795**	**0.8389**	0.4943	0.5735	**0.7959**
Chengdu	TMLKD w/o RK	0.2408	0.3094	0.5515	0.2453	0.3553	0.5997	0.3355	0.4068	0.6633	0.4053	0.4899	0.7122
	TMLKD w/o *s*	0.1846	0.2760	0.4690	0.1879	0.2908	0.4972	0.3074	0.4136	0.6552	0.3783	0.4887	0.7116
	TMLKD w/o Lcon	0.2408	0.3094	0.5515	**0.2585**	**0.3653**	0.6007	**0.4165**	**0.4796**	**0.7389**	**0.4113**	**0.4991**	0.7093
	TMLKD w/o Attn	0.2199	0.2913	0.5270	0.2275	0.3130	0.5838	0.3577	0.4351	0.7043	0.4036	0.4895	0.7025
	TMLKD	**0.2602**	**0.3509**	**0.5942**	0.2488	0.3634	**0.6049**	0.4136	0.4757	0.7299	0.4100	0.4982	**0.7193**

evaluation, TMLKD effectively avoids overfitting, thereby delivering more stable and superior performance in few-shot scenarios. To conclude, the possible reasons for the improved accuracy are as follows:

- TMLKD considers the domain shifts between different measures and transfers shareable representation knowledge. Besides, it distills transferable rank knowledge from teacher models. Hence, it mitigates the performance drop caused by sparse labels on the target measure compared to non-KD methods.
- TMLKD pays enough attention to the teacher-student consistency. When transferring knowledge from teacher models, TMLKD uses a sample-wise confidence evaluation to effectively differentiate high-quality teachers and attach more importance to it during the distillation. With the above improvements, TMLKD achieves the best performance.

5.3.2 Ablation Study (RQ3). We further perform ablation experiments to investigate the effect of the components of our method. Table 4 reports the results of all variant models.

Effect of Rank Knowledge. To investigate the effect of transferring rank knowledge from data-rich source measures, we design the variant *TMLKD w/o RK* by removing the rank knowledge distillation modules (including soft label enrichment and relaxed ranking loss). From the table, we can conclude that it is beneficial to conduct rank distillation, as it provides useful rank information for the student model and increases the accuracy.

Effect of Representation Knowledge. To investigate the effect of transferring domain-invariant representation to the student model, we design the following variants:

- *TMLKD w/o s*. We remove the domain-invariant representation provided by the teacher model and only use domain-specific trajectory representations.
- *TMLKD w/o Lcon*. We remove the orthogonality constraints and corresponding $L_{contrastive}$.

By integrating the shareable representation knowledge, TMLKD achieves a remarkable performance increase. For example, for the recall rate on the Porto dataset, TMLKD increases recall rate by 0.22 points under the SSPD measure. What is more, domain-specific representation, which is designed to extract scenario-oriented features, should be distinguished from domain-invariant representation. with the contrastive loss which prevents storing redundant knowledge. TMLKD further improves the model performance in most cases, which means that domain-specific representation preserves unique knowledge on the target measure effectively.

Effect of Attention Module. To investigate the impact of the attention module for domain-invariant representation knowledge extraction, we design the variant *TMLKD w/o Attn*, which removes the attention module during the training phase of teacher models. The performance of *TMLKD w/o Attn* falls behind that of TMLKD in most cases, indicating that the attention module assists the shareable knowledge transferring.

Table 5: Time Cost for Offline Model Training under SSPD distance on the Porto dataset.

Methods	n_{epoch}	t_{epoch}	t_{total}
NNS-SL	26.67 ± 2.31	88.43 ± 2.63s	2359.99 ± 243.37s
NNS-Non KD	77.67 ± 8.02	2.82 ± 0.29s	219.54 ± 36.02s
NNS-TMLKD	10.33 ± 1.15	8.13 ± 0.96s	83.56 ± 8.68s
T3S-SL	22 ± 7	190.64 ± 4.55s	2044.56 ± 3212.08s
T3S-Non KD	56.67 ± 2.89	8.2 ± 0.56s	464.21 ± 28.37s
T3S-TMLKD	63.67 ± 40.5	31.96 ± 3.4s	1943.89 ± 1127.77s

• t_{epoch} refers to the average training time of each epoch. n_{epoch} refers to the training epoch at which the model achieves the best performance on the validation set. t_{total} denotes the total training time. ± indicates the fluctuation of the evaluation metric across multiple tests.

Table 6: Time Cost for Similarity Computation of Different Methods on the Porto dataset (Unit: s).

Method	5k(5,000)	15k(15,000)	100k(100,000)
		SSPD	
Brute-Force	1897.2 ± 8.28	17142.09 ± 108.35	N/A
NNS-Non KD	6.26 ± 0.27	59.39 ± 1.70	5899.26 ± 24.09
NNS-TMLKD	6.24 ± 0.08	56.68 ± 1.35	5902.54 ± 21.42
T3S-Non KD	7.62 ± 0.21	69.67 ± 4.04	5928.54 ± 6.33
T3S-TMLKD	8.10 ± 0.42	63.54 ± 1.79	5940.82 ± 23.35
		Hausdorff	
Brute-Force	1884.8 ± 33.73	17051.82 ± 322.75	N/A
NNS-Non KD	6.12 ± 0.16	57.85 ± 0.28	5912.62 ± 24.99
NNS-TMLKD	5.98 ± 0.07	58.31 ± 4.71	5912.21 ± 21.31
T3S-Non KD	7.00 ± 0.12	57.84 ± 1.29	5901.24 ± 27.42
T3S-TMLKD	7.76 ± 0.18	61.99 ± 0.55	5906.70 ± 16.36
		DTW	
Brute-Force	2368.18 ± 13.87	21260.91 ± 161.38	N/A
NNS-Non KD	6.22 ± 0.18	58.64 ± 3.22	5914.22 ± 22.38
NNS-TMLKD	6.14 ± 0.19	55.61 ± 0.49	5884.45 ± 7.74
T3S-Non KD	6.88 ± 0.11	60.25 ± 1.99	5910.56 ± 35.9
T3S-TMLKD	7.64 ± 0.12	64.64 ± 0.52	5951.03 ± 49.17
		Discrete-Frechet	
Brute-Force	1864.27 ± 22.08	16801.09 ± 193.65	N/A
NNS-Non KD	6.00 ± 0.02	57.13 ± 1.29	5887.40 ± 27.13
NNS-TMLKD	6.17 ± 0.18	58.50 ± 4.89	5893.99 ± 29.45
T3S-Non KD	6.96 ± 0.12	59.44 ± 2.94	5882.61 ± 6.12
T3S-TMLKD	7.71 ± 0.11	63.14 ± 2.30	5946.91 ± 14.73

• N/A means the computation cannot be finished in a week

5.4 Efficiency Study (RQ4)

5.4.1 Time Cost of Offline Training. We first evaluate the training-time costs on Discrete-Frechet distance on the Porto dataset. The experiments are performed with three types of methods:

- **Model-SL**: We implement the base model with sufficient pair-wise distance labels.
- **Model-Non KD**: We implement the base model with insufficient relative ranking labels.
- **Model-TMLKD**: We improve the base model in data-sparse measure with our TMLKD method.

We report the experimental results in Table 5. When training Model-SL, the training phase involves more trajectories as the training and validation set, along with more annotations. Hence, each epoch requires more time. However, it is easy to converge (e.g. it only takes less than 30 epochs to achieve relatively ideal results). For the non-KD student model, since labels are insufficient, we only need roughly 20 minutes for the total training phase with T3S as the base model. Even if the training time of non-KD methods is short, their accuracy performance is not satisfactory.

For our TMLKD method, each epoch requires more time than non-KD method, since we introduce extra representation knowledge. What is more, non-KD methods only focus on the anchor trajectory and corresponding labeled trajectories, while TMLKD pays attention to unlabeled trajectories to enrich the sparse label, which results in extra time costs. Finally, the total training time of all the measures is less than 1 hour and roughly 20 minutes with T3S and NT-NNS as base models respectively, which is tolerable.

5.4.2 Time Cost of Distance Computation. Given several trajectories, we compare the time cost of similarity computation under different measures on the Porto dataset. The results are demonstrated in Table 6. During the evaluation, we randomly sample 5k, 15k and 100k trajectories from the Porto dataset. For the non-learning method (i.e. the Brute Force Method), we follow the formula of each measure according to its definition and compute the pair-wise trajectory distance. Overall, our method achieves over 200x speedup over the brute force method.

For learning-based methods, the computation time consists of two parts: embedding obtaining and pair-wise Euclidean distance calculation. Besides TMLKD methods, we test on Non-KD methods, which directly map trajectories into low-dimensional embeddings without distilling ranking and latent knowledge. According to Table 6, the calculation time of our method is a little longer than the base model. This is due to the utilization of the extra embedding module which introduces domain-invariant knowledge to the student model. However, this gap is tolerable, since the time difference is also within 100 seconds for 100k trajectories.

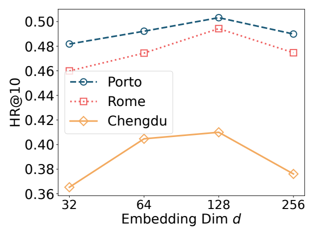
5.5 Parameter Sensitivity Study (RQ5)

To analyze the Impact of different parameters, we perform parameter sensitivity experiments on each dataset.

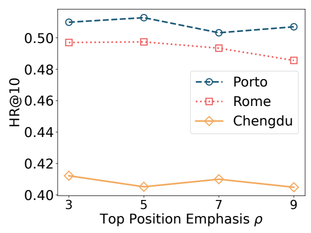
5.5.1 Impact of Embedding Dimension. We explore the impact of the embedding dimension d. It determines the performance of metric learning and influences the processing time of trajectories. As exhibited in Figure 3(a) and 4(a), we observe the model performances under 32, 64, 128, 256. It can be concluded from the figure that a small d is incapable of storing sufficient spatial and structural knowledge of trajectories. However, with a too large d, the model suffers from over-fitting problems and is unable to learn complex information from unnecessary dimensions. Finally, we choose 128 as the embedding dimension of our model.

5.5.2 Impact of Top position emphasis. We explore the impact of the Top Position Emphasis ρ. When ρ is small, the model pays more attention to trajectories at the top position. When ρ gets larger, the calculated correlated scores of trajectories at different positions show little variation. As exhibited in Figure 3(b) and 4(b), we observe the model performance under 3, 5, 7, 9. The results suggest that the overall performance is not highly sensitive to the choice of ρ, as all values yield comparable results. Nevertheless, we set ρ to 7 in our experiments, as it slightly improves performance by balancing the emphasis on top-ranked trajectories while still considering those at lower positions, which may also carry useful information.

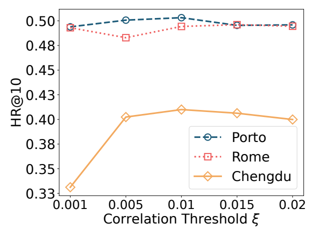
5.5.3 Impact of Correlation Threshold. We explore the impact of the Correlation threshold ξ. When ξ is small, more trajectories



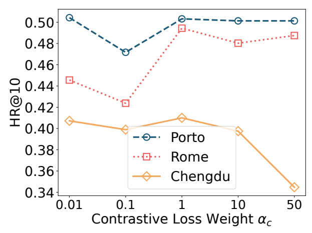
(a) Impact of d.



(b) Impact of ρ.

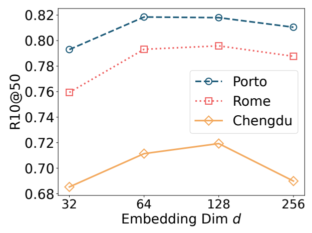


(c) Impact of ξ.

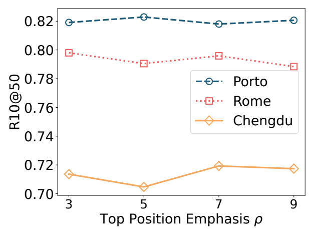


(d) Impact of α_c.

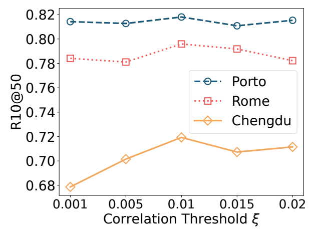
Figure 3: HR@10 of T3S-TMLKD under Discrete-Frechet distance on each dataset with varying embedding dimension d, top position emphasis ρ, correlation threshold ξ, contrastive loss weight α_c.



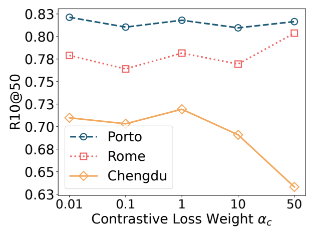
(a) Impact of d.



(b) Impact of ρ.



(c) Impact of ξ.



(d) Impact of α_c.

Figure 4: R10@50 of T3S-TMLKD under Discrete-Frechet distance on each dataset with varying embedding dimension d, top position emphasis ρ, correlation threshold ξ, contrastive loss weight α_c.

will be considered as weakly-correlated trajectories. As exhibited in Figure 3(c) and 4(c), we observe the model performance under 0.001, 0.005, 0.01.0.015, 0.02. It can be concluded from the figure that small ξ introduces too many trajectories as weakly correlated ones for anchor trajectory, which introduces the noisy rank knowledge at low positions and leads to performance drop. However, a larger ξ results in a small degradation as well, since it neglects similar trajectories that contain valuable information for the anchor trajectory. During the training procedure of the student model, we set ξ to 0.01 to effectively introduce rank knowledge from multiple teachers while avoiding misleading the student with noise.

5.5.4 Impact of Contrastive Loss Weight. Given a pair of trajectories, most base models minimize the difference of the pair-wise distance between trajectory embeddings and original trajectories. They use MSE as L_{base} in their framework, which is like our L_{share} and $L_{private}$. $L_{base}, L_{share} and L_{private}$ will not differ significantly in magnitude. Hence, α_s and α_p are set to 1 in our experiments when training teacher models. $L_{contrastive}$ uses orthogonality constraints to avoid duplicate information extracted by domain-invariant and domain-specific representations. As exhibited in Figure 3(d) and 4(d), we observe the performance of our model with the value of α_c set as 0.01, 0.1, 1, 10, 50 respectively. To conclude, an excessively large α_c may cause the model to ignore its own representation capability and metric performance, while an excessively small α_c may prevent the model from effectively incorporating the adversarial strategy for knowledge decoupling. Hence, we set α_c to 1 when training models.

6 CONCLUSION

In this paper, we present a few-shot trajectory metric learning model that addresses performance degradation caused by insufficient labels by distilling knowledge from teacher models trained with extensive annotations. To manage domain shifts when transferring knowledge across different measures, we employ adversarial shared-private sub-networks to separate domain-specific knowledge from domain-invariant information, enabling the transfer of shareable representations. This design ensures that the model adapts effectively to diverse data distributions while preserving critical shared features. To prevent the student model from being misled by low-quality teachers, we assess each teacher model's sample-wise confidence instead of fusing their outputs. Given the scarcity of labels in the target measure, we further apply a list-wise learning-to-rank approach to derive relaxed trajectory ranking orders from teachers' predictions, serving as soft labels to supplement limited labels. Extensive experiments on real-world datasets verify the effectiveness and efficiency of our proposed TMLKD.

ACKNOWLEDGMENT

The authors would like to thank the anonymous reviewers for their valuable comments. This work is supported by the National Natural Science Foundation of China (Grant No.62272334, No.61802273, No.61872258, No.61872258 and No.62102277), the Natural Science Foundation of Jiangsu Province (Grant No.BK20210703), the ARC Discovery Projects (Grant No.DP220102191 and No.DP200103700), the Jiangsu Higher Education Institutions of China (No.23KJA520011), and the Priority Academic Program Development of Jiangsu higher education institutions.

REFERENCES

[1] Helmut Alt. 2009. The Computational Geometry of Comparing Shapes. In *Efficient Algorithms, Essays Dedicated to Kurt Mehlhorn on the Occasion of His 60th Birthday (Lecture Notes in Computer Science)*, Susanne Albers, Helmut Alt, and Stefan Näher (Eds.), Vol. 5760. 235–248.

[2] Helmut Alt and Michael Godau. 1995. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geom. Appl.* 5 (1995), 75–91.

[3] Donald J. Berndt and James Clifford. 1994. Using Dynamic Time Warping to Find Patterns in Time Series. In *AAAI Workshop*, Usama M. Fayyad and Ramasamy Uthurusamy (Eds.). 359–370.

[4] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. 2016. Domain Separation Networks. In *NIPS*. 343–351.

[5] Wei-Lun Chang, Hui-Po Wang, Wen-Hsiao Peng, and Wei-Chen Chiu. 2019. All about structure: Adapting structural information across domains for boosting semantic segmentation. In *CVPR*. 1900–1909.

[6] Yanchuan Chang, Jianzhong Qi, Yuxuan Liang, and Egemen Tanin. 2023. Contrastive Trajectory Similarity Learning with Dual-Feature Attention. In *ICDE*. 2933–2945.

[7] Cen Chen, Chengyu Wang, Minghui Qiu, Dehong Gao, Linbo Jin, and Wang Li. 2021. Cross-domain knowledge distillation for retrieval-based question answering systems. In *WWW*. 2613–2623.

[8] Lei Chen and Raymond T. Ng. 2004. On The Marriage of Lp-norms and Edit Distance. In *VLDB*, Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer (Eds.). 792–803.

[9] Lei Chen, M. Tamer Özsu, and Vincent Oria. 2005. Robust and Fast Similarity Search for Moving Object Trajectories. In *SIGMOD*, Fatma Özcan (Ed.). 491–502.

[10] Wei Chen, Shuzhe Li, Chao Huang, Yanwei Yu, Yongguo Jiang, and Junyu Dong. 2022. Mutual Distillation Learning Network for Trajectory-User Linking. In *IJCAI*. 1973–1979.

[11] Connor Colombe and Kyle Fox. 2021. Approximating the (Continuous) Fréchet Distance. In *SoCG*, Vol. 189. 26:1–26:14.

[12] Songlin Dong, Xiaopeng Hong, Xiaoyu Tao, Xinyuan Chang, Xing Wei, and Yihong Gong. 2021. Few-Shot Class-Incremental Learning via Relation Knowledge Distillation. In *AAAI*. AAAI Press, 1255–1263.

[13] Ziquan Fang, Yuntao Du, Lu Chen, Yujia Hu, Yunjun Gao, and Gang Chen. 2021. E^2DTC: An End to End Deep Trajectory Clustering Framework via Self-Training. In *ICDE*. 696–707.

[14] Ziquan Fang, Yuntao Du, Lu Chen, Yujia Hu, Yunjun Gao, and Gang Chen. 2021. E2dtc: An end to end deep trajectory clustering framework via self-training. In *ICDE*. IEEE, 696–707.

[15] Takashi Fukuda, Masayuki Suzuki, Gakuto Kurata, Samuel Thomas, Jia Cui, and Bhuvana Ramabhadran. 2017. Efficient Knowledge Distillation from an Ensemble of Teachers. In *INTERSPEECH*. 3697–3701.

[16] Xudong Gong, Yan Xiong, Wenchao Huang, Lei Chen, Qiwei Lu, and Yiqing Hu. 2015. Fast Similarity Search of Multi-Dimensional Time Series via Segment Rotation. In *DASFAA*, Vol. 9049. 108–124.

[17] Golnaz Habibi and Jonathan P. How. 2021. Human Trajectory Prediction Using Similarity-Based Multi-Model Fusion. *IEEE Robotics Autom. Lett.* 6, 2 (2021), 715–722.

[18] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the Knowledge in a Neural Network. *CoRR* abs/1503.02531 (2015).

[19] Danlei Hu, Lu Chen, Hanxi Fang, Ziquan Fang, Tianyi Li, and Yunjun Gao. 2023. Spatio-Temporal Trajectory Similarity Measures: A Comprehensive Survey and Quantitative Study. *CoRR* abs/2303.05012 (2023).

[20] Sang-Yeong Jo and Sung Whan Yoon. 2023. POEM: Polarization of Embeddings for Domain-Invariant Representations. 37 (2023), 8150–8158.

[21] SeongKu Kang, Junyoung Hwang, Wonbin Kweon, and Hwanjo Yu. 2020. DE-RRD: A Knowledge Distillation Framework for Recommender System. In *CIKM*. 605–614.

[22] Jingzhi Li, Zidong Guo, Hui Li, Seungju Han, Ji-Won Baek, Min Yang, Ran Yang, and Sungjoo Suh. 2023. Rethinking Feature-based Knowledge Distillation for Face Recognition. In *CVPR*. 20156–20165.

[23] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S Jensen, and Wei Wei. 2018. Deep representation learning for trajectory similarity computation. In *ICDE*. IEEE, 617–628.

[24] Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2017. Adversarial Multi-task Learning for Text Classification. In *ACL*, Regina Barzilay and Min-Yen Kan (Eds.). 1–10.

[25] Yuang Liu, Wei Zhang, and Jun Wang. 2020. Adaptive multi-teacher multi-level knowledge distillation. *Neurocomputing* 415 (2020), 106–113.

[26] Alessio Monti, Angelo Porrello, Simone Calderara, Pasquale Coscia, Lamberto Ballan, and Rita Cucchiara. 2022. How many Observations are Enough? Knowledge Distillation for Trajectory Forecasting. In *CVPR*. 6543–6552.

[27] Luís Moreira-Matias, João Gama, Michel Ferreira, João Mendes-Moreira, and Luis Damas. 2016. Time-evolving OD matrix estimation using high-speed GPS data streams. *Expert systems with Applications* 44 (2016), 275–288.

[28] Jathushan Rajasegaran, Salman Khan, Munawar Hayat, Fahad Shahbaz Khan, and Mubarak Shah. 2021. Self-supervised Knowledge Distillation for Few-shot Learning. In *BMVC*. 179–192.

[29] Sayan Ranu, Padmanabhan Deepak, Aditya D Telang, Prasad Deshpande, and Sriram Raghavan. 2015. Indexing and matching trajectories under inconsistent sampling rates. In *ICDE*. IEEE, 999–1010.

[30] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. FitNets: Hints for Thin Deep Nets. In *ICLR*.

[31] Yasushi Sakurai, Masatoshi Yoshikawa, and Christos Faloutsos. 2005. FTW: fast similarity search under the time warping distance. In *ACM SIGACT-SIGMOD-SIGART*. 326–337.

[32] Han Su, Shuncheng Liu, Bolong Zheng, Xiaofang Zhou, and Kai Zheng. 2020. A survey of trajectory distance measures and performance evaluation. *VLDB J.* 29, 1 (2020), 3–32.

[33] Jiaxi Tang and Ke Wang. 2018. Ranking Distillation: Learning Compact Ranking Models With High Performance for Recommender System. In *SIGKDD*. 2289–2298.

[34] Raphael Tang, Yao Lu, Linqing Liu, Lili Mou, Olga Vechtomova, and Jimmy Lin. 2019. Distilling task-specific knowledge from bert into simple neural networks. *arXiv preprint arXiv:1903.12136* (2019).

[35] Lin Wang and Kuk-Jin Yoon. 2021. Knowledge distillation and student-teacher learning for visual intelligence: A review and new outlooks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 6 (2021), 3048–3068.

[36] Yuning Wang, Pu Zhang, Lei Bai, and Jianru Xue. 2023. Enhancing Mapless Trajectory Prediction through Knowledge Distillation. *CoRR* abs/2306.14177 (2023).

[37] Meng-Chieh Wu, Ching-Te Chiu, and Kun-Hsuan Wu. 2019. Multi-teacher Knowledge Distillation for Compressed Video Action Recognition on Deep Neural Networks. In *ICASSP*. 2202–2206.

[38] Zoe Xi and William Kuszmaul. 2022. Approximating Dynamic Time Warping Distance Between Run-Length Encoded Strings. In *ESA*, Vol. 244. 90:1–90:19.

[39] Chenxiao Yang, Junwei Pan, Xiaofeng Gao, Tingyu Jiang, Dapeng Liu, and Guihai Chen. 2022. Cross-Task Knowledge Distillation in Multi-Task Recommendation. In *AAAI*. 4318–4326.

[40] Peilun Yang, Hanchen Wang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. 2021. T3S: Effective Representation Learning for Trajectory Similarity Computation. In *ICDE*. 2183–2188.

[41] Ze Yang, Linjun Shou, Ming Gong, Wutao Lin, and Daxin Jiang. 2020. Model Compression with Two-stage Multi-teacher Knowledge Distillation for Web Question Answering System. In *WSDM*. 690–698.

[42] Di Yao, Gao Cong, Chao Zhang, and Jingping Bi. 2019. Computing Trajectory Similarity in Linear Time: A Generic Seed-Guided Neural Metric Learning Approach. In *ICDE*. 1358–1369.

[43] Di Yao, Haonan Hu, Lun Du, Gao Cong, Shi Han, and Jingping Bi. 2022. TrajGAT: A Graph-based Long-term Dependency Modeling Approach for Trajectory Similarity Computation. In *SIGKDD*. 2275–2285.

[44] Hanyuan Zhang, Xinyu Zhang, Qize Jiang, Baihua Zheng, Zhenbang Sun, Weiwei Sun, and Changhu Wang. 2020. Trajectory Similarity Learning with Auxiliary Supervision and Optimal Matching. In *IJCAI*. 3209–3215.