Weakly-supervised Temporal Path Representation Learning with Contrastive Curriculum Learning

Sean Bin Yang[1], Chenjuan Guo[1], Jilin Hu[1,✉] , Bin Yang[1], Jian Tang[2,3,4], and Christian S. Jensen[1]
[1]Department of Computer Science, Aalborg University, Denmark
[2]Mila-Quebec AI Institute [3]HEC Montreal, Canada [4]CIFAR AI Research Chair
{seany, cguo, hujilin, byang, csj }@cs.aau.dk, jian.tang@hec.ca

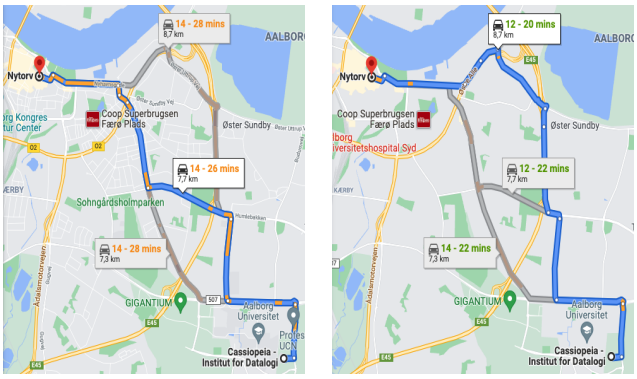
***Abstract*—In step with the digitalization of transportation, we are witnessing a growing range of path-based smart-city applications, e.g., travel-time estimation and travel path ranking. A temporal path (TP) that includes temporal information, e.g., departure time, into the path is of fundamental to enable such applications. In this setting, it is essential to learn generic temporal path representations (TPRs) that consider spatial and temporal correlations simultaneously and that can be used in different applications, i.e., downstream tasks. Existing methods fail to achieve the goal since (i) supervised methods require large amounts of task-specific labels when training and thus fail to generalize the obtained TPRs to other tasks; (ii) though unsupervised methods can learn generic representations, they disregard the temporal aspect, leading to sub-optimal results.**

To contend with the limitations of existing solutions, we propose a Weakly-Supervised Contrastive learning model. We first propose a temporal path encoder that encodes both the spatial and temporal information of a temporal path into a TPR. To train the encoder, we introduce weak labels that are easy and inexpensive to obtain, and are relevant to different tasks, e.g., temporal labels indicating peak vs. off-peak hour from departure times. Based on the weak labels, we construct meaningful positive and negative temporal path samples by considering both spatial and temporal information, which facilities training the encoder using contrastive learning by pulling closer the positive samples' representations while pushing away the negative samples' representations. To better guide the contrastive learning, we propose a learning strategy based on Curriculum Learning such that the learning performs from easy to hard training instances. Experimental studies involving three downstream tasks, i.e., travel time estimation, path ranking, and path recommendation, on three road networks offer strong evidence that the proposal is superior to state-of-the-art unsupervised and supervised methods and that it can be used as a pre-training approach to enhance supervised TPR learning.

I. Introduction

Road-network paths are central in many intelligent transportation system applications, such as path recommendation [1]–[3], routing [4]–[7], travel cost estimation [8]–[11], and traffic analysis [12]–[17]. Path representation (PR) learning is the process of learning representations of paths in the form of vectors with a fixed and relatively low dimensionality that is independent of the actual lengths of path. Thus, such representations can render downstream applications that operate on paths much more efficient than what is possible when operating directly on traditional, variable-length representations of paths. This illustrates the potential of path representation learning for improving intelligent transportation applications. Indeed, initial studies of path representation learning [18], [19] already exist.

Corresponding Author: Jilin Hu



(a) 8:00 a.m. (b) 10:00 a.m.

Fig. 1: Travel Time Estimation. Travel times of paths from *Cassiopeia* to *Nytorv* at (a) 8:00 a.m. and (b) 10:00 a.m.

In this study, we aim at learning generic temporal path representations (TPRs), meaning that the representations can be utilized in variety of downstream tasks, and we do so without the need for task-specific labeled data. Next the temporal aspect is essential in transportation applications. Consider the travel-time estimation example from Google Maps in Fig. 1. Travel from *"Cassiopeia"* to *"Nytorv"* takes longer at 8:00 a.m. than at 10:00 a.m., due to the traffic congestion during morning peak hours. Further, it can be seen that the path recommendation rankings are also different. It recommends to avoid the highway at 8:00 a.m. due to the heavy congestion there, while recommends the highway again at 10:00 a.m., when the traffic is clear. Learning path representations without considering the temporal aspect results in poor accuracy, which in turn reduces the utility of such representation in downstream tasks. However, it is non-trivial to learn generic TPRs using either supervised or unsupervised learning.

Supervised approaches (Fig. 2(a)) learn TPRs based on task-specific labels [20], [21]. We call these "strong" labels because their use targets specific tasks. For example, for the task of travel-time prediction, the time and path encoders first take as input a departure time t and a path p, respectively. Then, their outputs are aggregated into a travel-time (TT) specific TPR, which is then utilized to predict the travel time of path p when

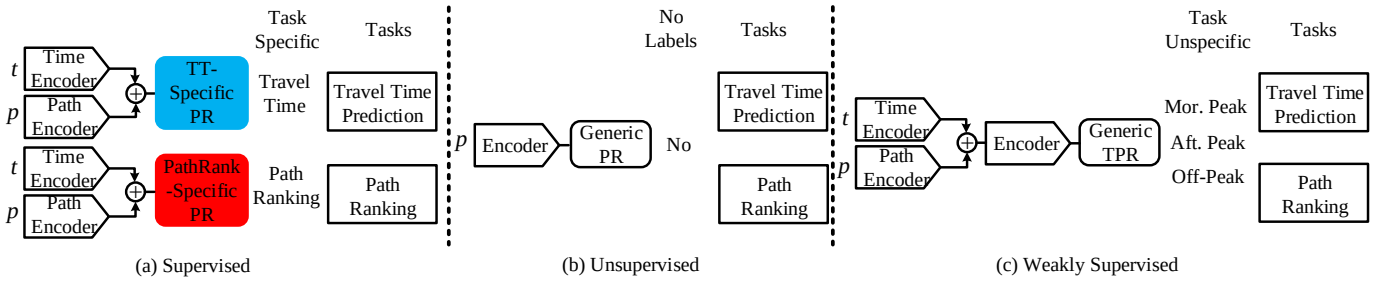


Fig. 2: Supervised, Unsupervised, and Weakly-Supervised methods for learning Temporal Path Representations (TPR): (a) Supervised learning relies on task-specific labels to obtain task-specific path representations (PRs), and thus fails to generalize across tasks; (b) Unsupervised learning produces generic path representations for use in different tasks, but fails to capture temporal traffic aspects of paths; (c) Weakly supervised learning (**Ours**) uses weak labels to learn generic TPRs.

departure at time t. This supervised learning approach has two drawbacks: (i) task specific TPRs do not generalize across tasks. For example, TPRs learned from travel-time labels may perform poorly in path ranking tasks. (ii) supervised learning requires a large amount of labeled training data, which may be impossible or expensive to obtain.

Unsupervised approaches do not rely on task specific labels and are thus able to offer generic path representations. Existing unsupervised path representation approaches rely heavily on unsupervised graph representation learning, where the representations of the edges in a path are aggregated into a path representation [18], [22]. Since existing graph representation learning does not consider temporal information, the obtained path representations also lacks the temporal aspect. However, as argued in the context of Fig. 1, disregarding the temporal aspect adversely affects the quality of downstream tasks.

In this paper, we target at a solution that is able to offer generic TPRs that take into account spatial and temporal correlations simultaneously without using task specific labels. To this end, we propose a temporal path encoder, consisting of a temporal and a spatial embedding module, to encode a temporal path into a TPR by considering both spatial and temporal information. Specifically, we construct a temporal graph to learn temporal embeddings for different departure times via graph representation learning; next, we embed various traffic related information from pertinent road networks into spatial embeddings. Finally, the temporal path encoder combines the temporal and spatial embeddings to generate the TPR of the input temporal path.

To enable the training of the temporal path encoder such that the obtained TPRs are generic and include temporal information, we introduce weak labels on the temporal aspect. Such weak labels are easy to obtain and are relevant to different tasks. Example weak labels include labels indicating peak vs. off-peak periods, which only depend on departure times. This way, all temporal paths are associated with weak labels according to their departure times.

Next, we construct meaningful positive and negative temporal path samples to enable contrastive learning such that no task-specific labels are required and thus the learned TPRs are generic across downstream tasks. The positive samples are those with the same paths and same weak labels and all other temporal paths, i.e., same paths with different weak labels, different paths with both same and different weak labels, are negative samples. To learn meaningful representations, we design an objective function to try to pull together representations of positive samples, while separating representations of negative samples. This enables generic TPRs while capturing the temporal information. Unlike the supervised methods, we therefore do not require strong, task-specific labeled data for our training. Instead, by deriving weak labels for temporal paths we obtain more generic representations. And unlike the unsupervised approaches, we consider the temporal aspect when we learn path representations.

To further enhance the weakly-supervised contrastive (WSC) learning, we integrate curriculum learning to improve the convergence rate and generalization capabilities of the TPR learning. Specifically, we propose an curriculum sample evaluation model that outputs difficulty scores for all training samples, according to which the training samples can be sorted. To achieve this we first split the training data set into non-overlapping meta-sets. Then, we train separate weakly supervised contrastive (WSC) models on each meta-set, respectively. Next, we calculate a TPR similarity score and treat it as a difficulty score for each training sample, based on which we sort all the samples. Finally, we provide a curriculum selection algorithm to perform the curriculum learning according to the difficulty scores.

To the best of our knowledge, this is the first solution that combines advantages of supervised and unsupervised learning to learn generic temporal path representations. In summary, we make the following contributions.

- We formulate the temporal path representation learning problem.
- We propose a weakly-supervised, contrastive model (basic framework) to learn generic path representations that take temporal information into account.
- We integrate curriculum learning into the weakly-supervised contrastive model to further enhance the learned temporal path representations, yields the advanced framework.
- We report on extensive experiments using three real-

world data sets on three downstream tasks to assess in detail the effectiveness of the proposed framework.

II. Related Work

A. Path Representation Learning

Deep learning is already being used for representation learning, and different studies have proposed a variety of methods to learn useful path representations. To the best of our knowledge, recurrent neural network (RNN) architectures, including long short-term memory (LSTM) [23] and gated recurrent unit (GRU) networks [24], have been established firmly as the state-of-the-art for path representation learning. Deepcas [25] leverages bi-directional GRUs to sequentially process forward and backward node representations of paths, representing a path by the concatenation of resulting forward and backward hidden vectors. ProxEmbed [26] uses LSTMs to process node representations and apply max-pooling on outputs across all time steps to generate a path representation. SPAE [21] proposes self-attentive path embedding. Paths of arbitrary length are first embedded into fixed-length vectors that are then fed to LSTMs to generate path representations. PathRank [20] propose a supervised path representation learning model that takes departure time as additional context information. The above methods all perform end-to-end training and rely on the availability of large amounts of labeled training data. In addition, their path representations are task specific. Most recently, the unsupervised path representation learning framework *PIM* [18] learns path representations. However, it does not include temporal information. In contrast, we propose a temporal path representation learning framework based on weakly-supervised contrastive loss that can learn path representations when given different departure times.

B. Contrastive Learning

Recently, the most effective approaches for learning representations with or without labeled data have been supervised or unsupervised contrastive learning [18], [22], [27]–[30], which have shown impressive performance in computer vision and graph learning. As a form of metric learning [31], contrastive approaches achieve representations in a discriminating manner through contrasting positive data pairs against negative data pairs. In early work, Hjelm et al. [27] proposed Deep InfoMax (DIM) for learning a generic image representations by maximizing mutual information between local and global features in an unsupervised manner. Inspired by DIM, Velickovic et al. [28] proposed a similar approach, called Deep Graph Informax (DGI), that learns graph-node representations in an unsupervised manner. Recently, Sun et al. [22] proposed InfoGraph for graph representation learning and evaluated the proposal in both unsupervised and semi-supervised settings. Most Recently, Khosla et al. [30] extended the self-supervised batch contrastive method to a fully-supervised setting, making it possible to leverage label information effectively. However, no previous studies have explored the direction of weakly supervised contrastive learning.

C. Curriculum Learning

Inspired by the human learning principle of starting by learning simple tasks before proceeding to learn increasingly hard tasks, curriculum learning (CL) [32] uses nonuniform sampling of mini-batches according to the order of sample difficulty. Due its great potential to improve sample efficiency for different deep learning models, CL has attracted considerable interest and has found application in different research domains, e.g., computer vision [33]–[35], and natural language processing (NLP) [36]–[38]. However, none of these studies apply CL to path representation learning. *PIM* [18] is the closet to our paper, in that it proposes a curriculum negative sampling method to enhance the path representation learning. However, *PIM* focuses on negative sampling generations, but not on training. Xu et al. [38] propose two-staged curriculum learning for NLP, including difficulty evaluation and curriculum arrangement. Inspired by Xu et al. [38], we propose a curriculum learning framework that can evaluate the difficulty levels of data in a training data set automatically, so that models can be trained on increasingly difficult subsets of the training data set. The new framework features two key novelties. (i) Difficulty score computation: In NLP settings, difficulty scores, e.g., accuracy or F1 score, are computed based on strong labels in a supervised setting. In contrast, our difficulty scores are computed based on representation similarities, which do not rely on strong labels. (ii) How the training data is split into metasets: In NLP settings, training data is often split into metasets at random. In contrast, we split the training data based on the lengths of paths. This facilitates distinguishing the difficulty scores of paths.

III. Preliminaries

We first cover important concepts and then present the problem statement.

A. Definitions

Definition 1: **Road network.** A road network is defined as a directed graph $G = (\mathbb{V}, \mathbb{E})$, where $\mathbb{V}$ is a set of vertices v_i that represent intersections and $\mathbb{E} \subset \mathbb{V} \times \mathbb{V}$ is a set of edges $e_i = (v_j, v_k)$ that represent edges. Fig. 3 shows an example road network.

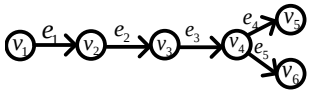


Fig. 3: An Example Road Network

Definition 2: **GPS Trajectory.** A GPS trajectory $traj = \langle (l_i, t_i) \rangle_{i=1}^{|traj|}$ of a moving object is defined as a timestamped location sequence, where l_i represents the GPS location at timestamp t_i.

Definition 3: **Path.** A path $\boldsymbol{p} = \langle e_i \rangle_{i=1}^{|\boldsymbol{p}|}$ is a sequence of adjacent edges, where $e_i \in \mathbb{E}$ is the i-th edge in the path.

Definition 4: **Temporal Path.** A temporal path is given by $tp = (\boldsymbol{p}, t)$, where $tp.\boldsymbol{p}$ is a path and $tp.\mathbf{t}$ is a departure time.

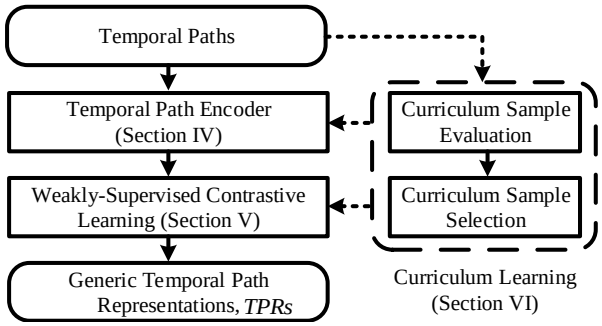


Fig. 4: Solution Overview

Definition 5: **Downstream Task.** A downstream task is a task that make estimations based on temporal path representations. Specifically, we consider travel time estimation, path ranking score estimation, and path recommendation.

Definition 6: **Weak Labels.** Weak labels are easy and inexpensive to obtain, which do not dependent on specific tasks but are relevant to different tasks.

Example. An example of weak labels are labels indicating peak v.s. off-peaks periods based on the departure time. For example, it can be Morning peak (7 to 9 a.m., weekdays), Afternoon peak (4 to 7 p.m., weekdays), and Off-peak (all other times). Such labels are easy and inexpensive to obtain, compared to task-specific labels, e.g., labels indicating travel time or path ranking for different paths. Meanwhile, such weak labels are also relevant for three downstream tasks, because the travel time, path ranking, and path recommendation of the same path during peak vs. off-peak periods often differ significantly.

Definition 7: **Temporal Path Representation.** The temporal path representation TPR_{tp} of a temporal path tp is a vector in $\mathbb{R}^{d_h}$, where d_h is the dimensionality of the vector.

B. Problem Statement

Given a set of temporal paths $\mathbb{TP} = \{tp_1, tp_2, \ldots, tp_n\}$ where each temporal path tp_i is with a weak label y_i, temporal path representation learning (*TPRL*) aims at learning a temporal path representation $TPRL(tp_i)$ for each temporal path $tp_i \in \mathbb{TP}$ as formulated in Eq. 1.

$$TPRL_{\psi}(tp_i) : \mathbb{R}^{d_{tem}} \times \mathbb{R}^{M \times d} \rightarrow \mathbb{R}^{d_h}, \tag{1}$$

where ψ represents the learnable parameters for the path encoder, M is the total number of edges in the path, d, d_{tem}, and d_h are the feature dimensions for an edge, a departure time embedding, and a resulted temporal path representation, respectively.

C. Solution Overview

Fig. 4 shows an overview of the proposed weakly-supervised contrastive curriculum learning (*WSCCL*), which consists of three modules: 1) Temporal path encoder, 2) Weakly-supervised contrastive learning, and 3) Curriculum Learning. The details of those modules are provided in Sections IV, V, and VI, respectively.

IV. Temporal Path Encoder

Fig. 5 gives an overview of the WSC base framework. We detail the Temporal Path Encoder, which consists of a Spatial Embedding layer, a Temporal Embedding layer, and an LSTM layer. Spatial embedding takes as input a sequence of edges and outputs a sequence of spatial feature representations. Temporal embedding takes temporal information as input and converts input to a temporal feature vector. Next, we concatenate the spatial and temporal vector representations and feed the resulting representation to the LSTM model that extracts coupled spatio-temporal relationships and outputs spatio-temporal edge representations. Finally, we aggregate these edge representations to obtain the desired temporal path representations.

A. Temporal Embedding

Motivated by Yuan et al. [39], we construct a temporal graph $G' = (\mathbb{V}', \mathbb{E}')$, where each node $v' \in \mathbb{V}'$ denotes a departure time slot and each edge $e' \in \mathbb{E}'$ denotes a connection between two time slots. We first split the 24 hours of a day into 5-minute time slots to get 288 time slots. Then, to capture periodicities, we consider the 7 days of a week separately to get a total of 2016 nodes in the temporal graph, where each node represents a time slot and a day of the week. Next, we use two one-hot vectors, $\boldsymbol{t}_s \in \mathbb{R}^{288}$ and $\boldsymbol{t}_w \in \mathbb{R}^{7}$, to denote the initial representations for time slots and days of the week, respectively. For example, the departure time 00:06 a.m. on Monday is represented as $\boldsymbol{t}_s = [0, 1, 0, \cdots, 0]$ and $\boldsymbol{t}_w = [1, 0, 0, 0, 0, 0, 0]$.

Therefore, the node representation $\boldsymbol{t}_g^{emb}$ of temporal graph G' can be formulated as the concatenation of the two representations: $\boldsymbol{t}_g^{emb} = [\boldsymbol{t}_s, \boldsymbol{t}_w] \in \mathbb{R}^{288+7}$.

To consider the local similarities and weekly periodicities, we draw connections between different nodes in the temporal graph. More specifically, we connect adjacent time slots, indicating that neighboring time ranges should be similar. Further, we connect the adjacent nodes during neighboring days, indicating that time ranges during neighboring days should be similar. Finally, we also connect the time slots between Sunday and Monday.

Next, we apply a graph representation model, specifically, *node2vec* [40], to the temporal graph to further learn node representations in the graph. This is expressed in Eq. 2.

$$\boldsymbol{t}^{all} = \text{Node2Vec}^{tg}(\boldsymbol{t}_g^{emb}), \tag{2}$$

where $\boldsymbol{t}^{all} \in \mathbb{R}^{d_{tem}}$ is the finalized temporal representation.

B. Spatial Embedding

Recall that a path consists of a sequence of edges, each of which as a number of spatial features, including, e.g., road types, number of lanes.

a) Capture of Spatial Features: The intuitive way to learn TPRs is to encode the spatial features of all edges in a temporal path into TPRs. We consider the following four types of spatial edge features: *Road Type (RT):* a categorical value that includes primary, secondary, residential, etc. *Number of*

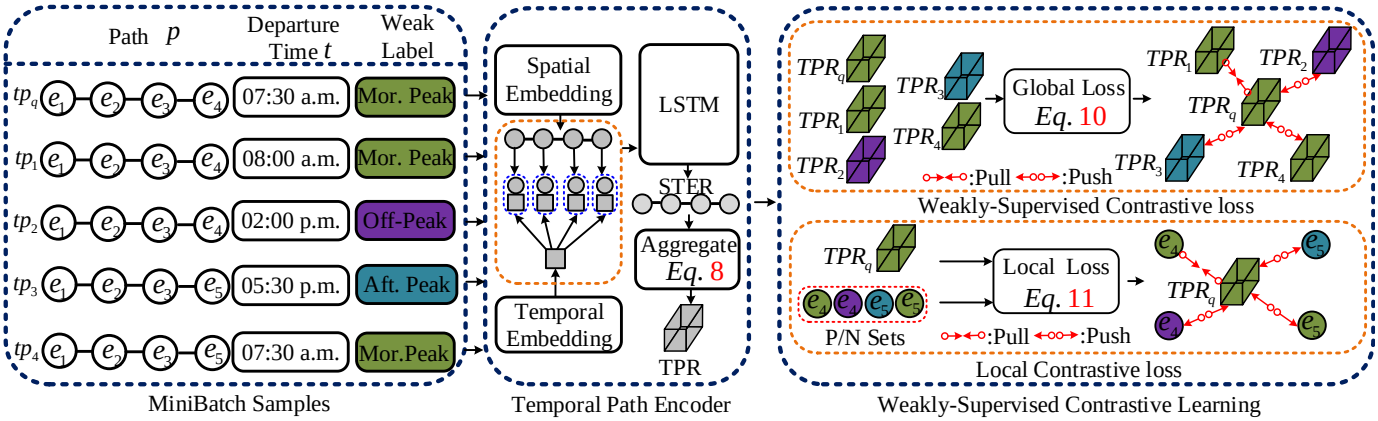


Fig. 5: Illustration of Basic Framework. Given a set of temporal paths in a minibatch, an input temporal path is encoded into a feature map by the temporal path encoder. The global loss framework takes a (query temporal path tp_q, positive or negative temporal path) pair as input and pulls together the TPRs of the query path and positive path, while pushing away the TPRs of the query path and negative paths. The local loss takes as input a TPR of query path and a spatio-temporal edge representation (STER) and brings a TPR of a query path with the positive edge representations closer while pushing apart TPR of a query path with the negative edge representations.

Lanes (NoL): a real value that represents the number of traffic lanes in the edge. *One Way (OW):* a Boolean that indicates whether the edge is one way or not. *Traffic Signals (TS):* a Boolean that indicates whether the edge has one or more traffic signals on the edge or not.

Next, we represent these different categorical features as one-hot vectors, which can be formulated as $\boldsymbol{s}_{RT}^{one} \in \mathbb{R}^{n_{rt}}$, $\boldsymbol{s}_{NoL}^{one} \in \mathbb{R}^{n_l}$, $\boldsymbol{s}_{OW}^{one} \in \mathbb{R}^{n_o}$, $\boldsymbol{s}_{TS}^{one} \in \mathbb{R}^{n_{ts}}$, where n_{rt}, n_l, n_o, n_{ts} represent the number of possible values in the four types of features.

Afterward, we leverage embedding matrices to convert these sparse one-hot vectors into dense vectors, which is formulated in Eq. 3.

$$\begin{aligned} \boldsymbol{s}_{RT}^{emb} &= \boldsymbol{M}_{RT}^T \times \boldsymbol{s}_{RT}^{one}\ ,\ \boldsymbol{s}_{NoL}^{emb} = \boldsymbol{M}_{NoL}^T \times \boldsymbol{s}_{NoL}^{one}, \\ \boldsymbol{s}_{OW}^{emb} &= \boldsymbol{M}_{OW}^T \times \boldsymbol{s}_{OW}^{one}\ ,\ \boldsymbol{s}_{TS}^{emb} = \boldsymbol{M}_{TS}^T \times \boldsymbol{s}_{TS}^{one}, \end{aligned} \tag{3}$$

where $\boldsymbol{M}_{RT} \in \mathbb{R}^{n_{rt} \times d_{rt}}$, $\boldsymbol{M}_{NoL} \in \mathbb{R}^{n_l \times d_l}$, $\boldsymbol{M}_{OW} \in \mathbb{R}^{n_o \times d_o}$, and $\boldsymbol{M}_{TS} \in \mathbb{R}^{n_{ts} \times d_{ts}}$, where d_{rt}, d_l, d_o, and d_{ts} are feature dimensions for dense vectors of RT, NoL, OW, and TS, respectively.

Finally, we concatenate all four dense features as final spatial feature embeddings for edge s_i, which can be formulated as follows.

$$\boldsymbol{s}^{type} = [\boldsymbol{s}_{RT}^{emb}, \boldsymbol{s}_{NoL}^{emb}, \boldsymbol{s}_{OW}^{emb}, \boldsymbol{s}_{TS}^{emb}], \tag{4}$$

where $[\cdot,\cdot]$ denotes concatenation vectors.

b) Road Network Topology: Since each edge has effects on its neighboring edges, connected edges should have similar representations. Inspired by graph embedding [28], [29], [40], which aims to learn node representations in a graph by considering the graph topology. Again, we apply *node2vec* [40] to learn graph representation of road network, which can be formulated as $\boldsymbol{n}_{vi}^{rn} = \text{Node2Vec}^{rn}(\boldsymbol{n}_{vi}^{one})$, where $\boldsymbol{n}_{vi}^{one}$ is one-hot vector of node v_i and $\boldsymbol{n}_{vi}^{rn} \in \mathbb{R}^{\frac{d_{top}}{2}}$ is the finalized node representation in a road network. The finalized edge representation of $e_k \in \mathbb{E}$ can be formulated as follows.

$$\boldsymbol{s}_{e_k}^{rn} = [\boldsymbol{n}_{v_i}^{rn}, \boldsymbol{n}_{v_j}^{rn}], \tag{5}$$

where v_i and v_j are start and end nodes of edge e_k.

Finally, the finalized topology feature with spatial feature of edge e_k can be rewritten as follows.

$$\boldsymbol{s}_{e_k}^{all} = [\boldsymbol{s}_{e_k}^{rn}, \boldsymbol{s}^{type}], \tag{6}$$

where $\boldsymbol{s}_{e_k}^{all} \in \mathbb{R}^d$ is the finalized spatial embedding for e_k, where $d = d_{rt} + d_l + d_o + d_{ts} + d_{top}$.

C. LSTM Encoder

Given an input temporal path tp, we achieve a sequence of spatio-temporal representations $\langle \boldsymbol{x}_{e_1}, \boldsymbol{x}_{e_2}, \cdots \boldsymbol{x}_{e_p} \rangle$, where $\boldsymbol{x}_{e_i} = [\boldsymbol{t}^{all}, \boldsymbol{s}_{e_i}^{all}]$, where $\boldsymbol{t}^{all}$ is the temporal embedding for $tp.t$. As recurrent neural networks (RNN) are known to be effective at modelling sequences, we feed this spatio-temporal representation into an RNN to further learn path representations. Specifically, we employ an LSTM model [41] to capture the sequential dependencies by taking each element of the spatio-temporal representations as input, which can be formulated as follows.

$$\hat{\mathbf{p}} = \langle \vec{p}_{e_1}, \vec{p}_{e_2}, \cdots \vec{p}_{e_p} \rangle = \text{LSTM}(\langle \boldsymbol{x}_{e_1}, \boldsymbol{x}_{e_2}, \cdots \boldsymbol{x}_{e_p} \rangle), \tag{7}$$

where $\vec{p}_{e_j} \in \mathbb{R}^{d_h}$ is the finalized spatio-temporal representation of edge e_j. LSTM$(\cdot)$ is the RNN model to capture the sequential dependencies, but it is also possible to use more advanced sequential models, e.g., Transformer [42].

D. Aggregate Function

As shown in Fig. 5, the *Aggregate* function takes as input a sequence of spatio-temporal edge feature vectors and returns a TPR. In particular, we aggregate $\hat{\mathbf{p}}$ into a TPR via an aggregate function $\mathcal{P}(\cdot) : \mathbb{R}^{n \times d_h} \longmapsto \mathbb{R}^{d_h}$, where n is the number of

edges in the path. We use an average aggregate function that takes the average of the edge representations in $\hat{\mathbf{p}}$ across edges.

$$\vec{h}_p = \frac{\sum_{i=1}^{n} \vec{p}_i}{|\hat{\mathbf{p}}|} \in \mathbb{R}^{d_h}, \tag{8}$$

where $\vec{h}_p$ represents the temporal path representation. $\vec{p}_i$ is the latent representation of edge i in the path.

V. Weakly-supervised Contrastive Learning

To ensure that we obtain generic TPRs that apply to different downstream tasks, we employ contrastive learning to construct the learning objectives for the whole framework. Here, we first detail positive and negative sample generation with weak labels. Then, we show how to construct weakly-supervised contrastive global and local losses.

A. Generation of Positive and Negative Samples

Positive and negative samples play an essential role in contrastive learning. Contrastive learning does not require strong labels and provides us with a good way of constructing the learning objectives for our model. In self-supervised contrastive learning, positive samples are always derived from the same object with different views, e.g., cropped parts of an object or generated by different models. Negative samples are simply representations that come from different objects. However, if some negative samples that have properties that are similar to those of positive samples, self-supervised contrastive learning faces difficulties because all negative samples are treated equally.

Suppose we have a set of temporal paths, positive TPs are not only different representations of the same temporal path, but they also include TPs that traverse the same path with the same weak label. In contrast, negative TPs belong to three categories: (i) same paths but different weak labels; (ii) different paths but the same weak labels; (iii) different paths and different weak labels. Therefore, we can generate multiple positive and negative TPs for a query TP.

The block of MiniBatch Samples in Fig. 5 shows an example, with five TPs, i.e., $tp_q, tp_1, tp_2, tp_3,$ and tp_4, and three weak labels, i.e, morning peak (Mor. Peak), Afternoon peak (Aft. Peak) and Off-Peak. If we take tp_q as the query TP, tp_1 is the corresponding positive sample since the two share the same path (i.e., $\langle e_1, e_2, e_3, e_4\rangle$) and the same departure weak label (i.e., Mor. Peak), although their exact departure times are different. Next, $tp_2, tp_3,$ and tp_4 are negative samples, where tp_2 has the same path but a different weak label, tp_3 has a different path and a different weak label, and tp_4 has a different path but the same weak label.

B. Global Weakly-supervised Contrastive Loss

Given a batch of training samples with batch size B, self-supervised contrastive loss can be formulated as in Eq. 9.

$$\mathcal{L}^{\text{self}} = \sum_{i=1}^{2B} -\log \frac{\exp\left(< \Psi\left(x_i'\right), \Psi\left(x_p'\right) > /\Psi\left(x_k'\right) > /\hat{\tau}\right)}{\sum_{k=1}^{2B} \mathbf{1}_{i\neq k} \exp < \left(\Psi\left(x_i'\right), \Psi\left(x_k'\right) > /\hat{\tau}\right)}, \tag{9}$$

where $\exp(\cdot)$ denotes the exponential operation, $< \cdot, \cdot >$ denotes the inner product of two vectors, $\Psi(\cdot) \in \mathbf{R}^{d_h}$ represents the output from the encoder, $\hat{\tau}$ is a temperature parameter, x_p' is an alternative view of object x_i' which can be generated by using data augmentation, e.g., rotate an image by 90 degree. x_k' denotes the negative samples from the batch, which is all samples from the batch other than x_i', meanwhile $\mathbf{1}_{i\neq k}$ is an indicator vector, where all elements are 1s, except a 0 at the i-th position.

However, the self-supervised contrastive loss in Eq. 9 is unable to take into account differences among negative samples. Motivated by *SupCon* [30], good generalization requires the ability to capture the similarity between samples in the same class and contrast them with samples in other classes. In this paper, we propose instead a WSC loss that utilizes positive and negative sample generation, as introduced in Section V-A. As is shown in Fig. 5, for each query temporal path tp_q, we try to pull closer TPRs with positive temporal path samples, which can be represented as $TPR_q \rightarrow\leftarrow TPR_1$, and push away TPRs from negative temporal path samples, denoted by $TPR_q \longleftrightarrow TPR_2$, $TPR_q \longleftrightarrow TPR_3$, and $TPR_q \longleftrightarrow TPR_4$. This yields global WSC formulated in Eq. 10.

$$\mathcal{L}^{global} = \sum_{(\boldsymbol{tp}_i, \boldsymbol{y}_i)\in\mathbb{P}} \mathcal{L}^{global}_{(\boldsymbol{tp}_i, \boldsymbol{y}_i)} = \sum_{(\boldsymbol{tp}_i, \boldsymbol{y}_i)\in\mathbb{P}} \frac{1}{|\mathbb{S}_{\boldsymbol{tp}_i}|} \sum_{\boldsymbol{tp}_j\in\mathbb{S}_{\boldsymbol{tp}_i}} \log \frac{\exp\left(\text{sim}(TPR_i, TPR_j)\right)}{\sum_{\boldsymbol{tp}_k\in\mathbb{N}_{tp_i}} \exp\left(\text{sim}(TPR_i, TPR_k)\right)}, \tag{10}$$

where $\text{sim}(\cdot)$ is cosine similarity function that quantifies the similarity between two TPRs; $\mathbb{P} = \{(tp_i, y_i)\}_{i=1}^{B}$ is a set of temporal paths in one training batch, where y_i is the departure weak label for tp_i; $\mathbb{S}_{tp_i} = \{tp_j\}_{j=1}^{|\mathbb{S}_{tp_i}|}$ is the positive sample set for query tp_i, where $y_i = y_j$ and $tp_i.\mathbf{p} = tp_j.\mathbf{p}$; and $\mathbb{N}_{tp_i} = \mathbb{P} \setminus \{\boldsymbol{tp}_i \cup \mathbb{S}_{\boldsymbol{tp}_i}\}$ is the negative sample set for query tp_i.

C. Local WSC Loss

In addition to the weakly-supervised learning across query temporal path with global positive and negative temporal paths, we also consider local differences between positive and negative temporal edge samples. These acts as a strong regularization that enhances the learning ability of our method. The Local Contrastive Loss element in Fig. 5 illustrates the design of our local contrastive loss, which consists of WSC with TPR_{tp_q}.

The goal of local contrastive learning is to preserve the local similarity between a TPR and the spatio-temporal representation of its edges. In particular, it is expected that a TPR can capture local similarity (edge-level similarity), i.e., TPRs are close to embeddings of positive edge samples and are distant from embeddings of negative edge samples. Similar to global WSC, we formulate the local WSC loss as maximizing the similarity with positive temporal edge samples as well as minimizing the similarity with negative temporal edge samples.

We proceed to describe the construction of the positive and negative edge samples. First, we randomly select edges that appear in positive temporal paths as our positive edge set, which is denoted as $\mathbb{PN}_i$. Then, edges that appear in negative temporal paths are selected as our negative edge set, denoted as $\mathbb{NN}_i$. Next, we set the weak temporal label of each temporal path be the label of the corresponding edge in the temporal path. As is shown in Fig. 5, for each query path tp_q, we try to pull TPRs with positive edges closer, e.g., (TPR_q, Mor. Peak) $\rightarrow\leftarrow$ (*STER*(e_4), Mor. Peak), and push away TPRs from negative edges, e.g., (TPR_q, Mor. Peak) $\leftarrow\rightarrow$ (*STER*(e_4), Off-Peak), (TPR_q, Mor. Peak) $\leftarrow\rightarrow$ (*STER*(e_5), Aft. Peak), and (TPR_q, Mor. Peak) $\leftarrow\rightarrow$ (*STER*(e_5), Mor. Peak).

In this phase, the objective of local contrastive learning is to increase the similarity of TPRs with positive edge samples while decreasing the similarity of TPRs with negative edge samples. Using cosine similarity $s(x, y) = x^\top y/\|x\|\|y\|$, we aim to optimize the objective function for local contrastive loss that is formulated in Eq. 11.

$$\mathcal{L}^{local} = \sum_{(\boldsymbol{tp}_i, \boldsymbol{y}_i)\in\mathbb{P}} \frac{1}{|\mathbb{PN}_i|} \log \frac{\sum_{(\boldsymbol{e}_j, y_i)\in\mathbb{PN}_i} \exp\left(s(TPR_i, e_j)\right)}{\sum_{(\boldsymbol{e}_k, y_j\neq y_i)\in\mathbb{NN}_i} \exp\left(s(TPR_i, e_k)\right)}, \tag{11}$$

where $\mathbb{PN}_i$ and $\mathbb{NN}_i$ are the positive and negative edge sets, and y_i denotes the weak label for edge representation, which inherits from the corresponding temporal path.

D. Objective for WSC

To train our temporal path encoder in an end-to-end manner, we jointly leverage both the the weakly-supervised global and local contrastive loss. Specifically, the overall objective function to maxmize is defined in Eq. 12.

$$\mathcal{L} = \arg\max_{\psi} \sum_{i\in I} \lambda \cdot \mathcal{L}_i^{global} + (1-\lambda)\cdot \mathcal{L}_i^{local}, \tag{12}$$

where λ is a balancing factor and I is the set of training batches.

VI. Contrastive Curriculum Learning

When training WSC using randomly shuffled training data, the training process is prone to getting stuck in a bad local optimum, which leads to suboptimal TPRs. To alleviate this problem, we build on the intuition that the algorithm should be presented with the training data in a meaningful order that facilitates learning. Specifically, the order of the samples is determined by how easy they are, as this can be expected to enhance weakly-supervised contrastive learning. We proceed to integrate curriculum learning with WSC, thus obtaining the advanced framework called *WSCCL*.

A. Overview of Curriculum Learning

Motivated by Xu et al. [38], we decompose curriculum sample generation into two stages: *Curriculum Sample Evaluation* and *Curriculum Sample Selection*, as shown in Fig. 6.

1) In curriculum sample evaluation, we assign a difficulty score S_i to path p_i in the training dataset $\mathbb{D}$. The score reflects the difficulty of the model to learn a good representation w.r.t. path p_i. 2) In curriculum sample selection, we aim to partition the training data into different difficulty stages. More specifically, we first sort the training data according to the difficulty scores. Then, we split the sorted training data into a sequence of sorted learning stages $\{ST_i|i = 1, 2, \ldots, M\}$ in an easy-to-difficult fashion. Finally, our base model, WSC, is trained according to this curriculum. We detail these two stages in Sections VI-B and VI-C, respectively.

B. Curriculum Sample Evaluation

The difficulty of a given temporal path can be quantified in many different ways. We argue that difficulty scores, like the intrinsic properties of the training dataset, should be decided by the model itself.

We first sort the training data set $\mathbb{D}$ according to the lengths of the paths. Then, we split $\mathbb{D}$ into N non-overlapping meta-sets, i.e., $\mathbb{D} = \{\widetilde{\mathbb{D}}_1, \widetilde{\mathbb{D}}_2, \cdots, \widetilde{\mathbb{D}}_N\}$, where $\widetilde{\mathbb{D}}_i$ is the i-th meta-set, $\widetilde{\mathbb{D}}_i \cap \widetilde{\mathbb{D}}_j = \emptyset, \forall i \neq j, i, j \in [1, N]$. Next, we train N independent WSC models w.r.t. the different meta-sets, i.e., $\widetilde{WSC}_i, i = [1, N]$. More specifically, $\widetilde{WSC}_i$ is trained only on the i-th meta-set, $\widetilde{\mathbb{D}}_i$. After that, we obtain a total of N trained independent WSC models, which we call *Experts*, to evaluate the difficulty of each training sample.

We then use the *Experts* to calculate the difficulty scores for the training data. We take a temporal path $\boldsymbol{tp}_i \in \widetilde{\mathbb{D}}_j$ as input to each *Expert*, $\widetilde{WSC}_j, j \in [1, N]$, and obtain a total of N TPRs, i.e., $TPR^{(1)}_{\boldsymbol{tp}_i}, TPR^{(2)}_{\boldsymbol{tp}_i}, \cdots, TPR^{(N)}_{\boldsymbol{tp}_i}$. Since $\boldsymbol{tp}_i$ comes from $\widetilde{\mathbb{D}}_j$, we take $TPR^{(j)}_{\boldsymbol{tp}_i}$ as the ground truth, and calculate the similarity between $TPR^{(j)}_{\boldsymbol{tp}_i}$ and $TPR^{(k)}_{\boldsymbol{tp}_i}$, where $\forall k \neq j, k \in [1, N]$. The resulting similarity scores are then summed up to denote as the difficulty score of temporal path $\boldsymbol{tp}_i$, denoted by S_i. This is formulated in Eq. 13.

$$S_i = \sum_{k=1, k\neq j}^{N} Sim(WSC_j(tp_i), WSC_k(tp_i)), \tag{13}$$

where $Sim(\cdot, \cdot)$ denotes the similarity function.

Finally, we repeat the difficulty score calculation for the entire training data. This yields a temporal path dataset $\mathbb{D} = \{(\boldsymbol{tp}_1, S_1), \cdots, (\boldsymbol{tp}_N, S_N)\}$, where $(\boldsymbol{tp}_i, S_i)$ is one element and S_i is the difficulty score for temporal path $\boldsymbol{tp}_i$. The intuition is that if one element, i.e., $(\boldsymbol{tp}_i, S_i)$ can obtain a similar representations in all other *Experts* compared to the representation from its own *Expert*, then we denote it as an easy sample. Since we calculate the sum of similarities, the higher S_i is, the easier the temporal path $\boldsymbol{tp}_i$.

C. Curriculum Sample Selection

To define the curriculum sample selection strategy, we first represent the learning curriculum in a multi-stage manner: $\{ST_i|i = 1, 2, \ldots, M\}$. More specifically, we rank all the training samples according to their difficulty scores and then distribute them evenly among M training stages. This way, the training data is partitioned into M parts with different levels of difficulty, ranging from S_1 (the easiest) to S_M (the

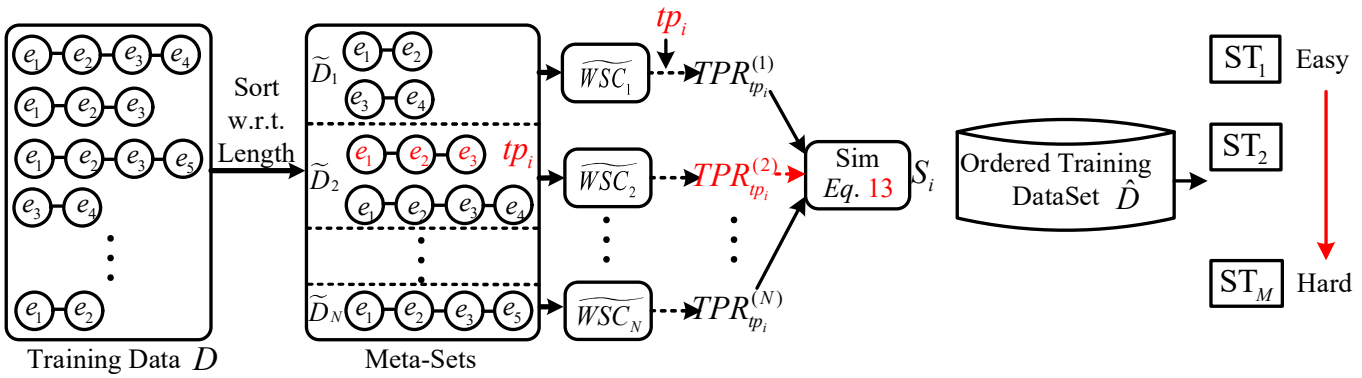


Fig. 6: Illustration of Advanced Framework

hardest). To ensure some local variations, the training samples in different stages are shuffled.

Next, we train our WSC for one epoch at each stage. When the training reaches stage S_M, the WSC should be ready for the original distribution in the whole training dataset $\mathbb{D}$. Finally, we add another stage, denoted by S_{M+1}, that covers the entire training data, and WSC is trained on this stage until it converges. For simplicity, we keep $N = M$, as is done elsewhere [38], and we leave the investigation of different combinations of N and M for future work.

VII. Experimental Study

A. Experimental Setup

1) Data sets: We use three traffic data sets to study the effectiveness of the proposed framework. Using each of these, we report results for three downstream tasks: travel time prediction, path ranking, and path recommendation.

a) Aalborg, Denmark [20]: We use the road network graph of Aalborg from OpenStreetMap[1] that consists 10,017 nodes and 11,597 edges. We use a substantial GPS data set that captures travel in this road network. Specifically, the data set consists of 180 million GPS records from 183 vehicles sampled at 1 Hz over a two-year period. A well-known map matching method [43] is used to map match the trajectories. This yields a total of 28,370 paths.

b) Harbin, China [19]: The data set was collected from 13,000 taxis in Harbin, China. We extract the corresponding road network from OpenStreetMap. The network contains 8,497 nodes and 14,497 edges. The GPS data was sampled at about 1/30 Hz. After map matching, we obtain 58,977 paths.

c) Chengdu, China[2]: The data set was collected in Chengdu, China during October and November 2016. We extract the corresponding road network from OpenStreetMap. The network contains 6,632 nodes and 17,038 edges. The GPS data was sampled at about 1/4–1/2 Hz. After map matching, we obtain 57,404 paths.

2) Downstream Tasks: We consider three tasks.

a) Path Travel Time Estimation: Each path is associated with a travel time (in seconds) obtained from the corresponding trajectory. We aim at building a regression model to estimate the travel times of paths based on their TPRs. We evaluate the accuracy of estimations using the Mean Absolute Error (**MAE**), Mean Absolute Relative Error (**MARE**), and Mean Absolute Percentage Error (**MAPE**). Smaller values indicate higher estimation accuracy.

b) Path Ranking: In path ranking, each path is associated with a ranking score in the range $[0, 1]$. The ranking scores are obtained with the help of historical trajectories by following an existing procedure [20]. Given a historical trajectory of a driver, we consider the path used by the trajectory, called the trajectory path, as the top ranked path. Then, we use a path finding algorithm to generate multiple paths connecting the same source and destination. We use the similarity between a generated path and the trajectory path for ranking: the more similar a generated path is to the trajectory path, the higher its similarity score; and the trajectory path itself has score 1 and thus ranks the highest. As for the previous task, we aim at building a regression model to estimate the ranking scores of paths.

To quantify the performance of path ranking, we report the **MAE** of the estimated ranking scores, the Kendall rank correlation coefficient (τ) [44], and the Spearman's rank correlation coefficient (ρ) [45]. The latter two capture the similarity, or consistency, between the ground truth and estimated rankings. The higher the τ and ρ values are, the more consistent the two rankings are, indicating higher accuracy.

c) Path Recommendation: A similar strategy is used in an existing study [46], where a path is associated with a binary label with the help of users' trajectories. A path used by a user's trajectory, say path A, is labeled 1, whereas alternative paths connecting the same source and destination, say paths B and C, are labeled 0. This follows the intuition that given three paths A, B, and C, the user should choose A, not B and C, meaning that path A should be recommended to the user. We conduct the path recommendation task on all three data sets. We evaluate the recommendation effectiveness using

[1]https://www.openstreetmap.org
[2]https://outreach.didichuxing.com/research/opendata/en/

the classification **Accuracy** (Acc.) and **Hit Rate** (HR). Higher values indicate better performance.

3) Baselines: We compare *WSCCL* with 12 baseline methods, which include 7 unsupervised methods and 5 supervised methods. The unsupervised methods are: (1) **Node2vec** [40], Deep Graph InfoMax (**DGI**) [28], Graphical Mutual Information Maximization (**GMI**) [29] are unsupervised graph representation learning frameworks, that give the edge representation for each edge in a graph. We use the average of the edge representations of the edge in a path as the path's representation. (2) **Memory Bank (MB)** [47] is an unsupervised learning approach to learn representations based on contrastive loss, where the representations of negative sample are randomly selected from memory bank. We re-implement **MB** with an LSTM encoder to capture the sequential information in paths. (3) **InfoGraph** [22] is a graph representation learning framework for unsupervised and semi-supervised settings. Here, we consider the unsupervised variant and treat a path as a graph to learn the path's representation. (4) **BERT** [48] is an unsupervised language representation learning model. To enable training, we treat a path as a sentence and mask some edges in the path. Then we split a path P into sub-paths P_1 and P_2, and consider (P_1, P_2) as a valid question-answer (Q&A) pair and (P_2, P_1) as an invalid Q&A pair because the former represents a meaningful ordering while the latter does not. (5) **PIM** [18] is an unsupervised path representation learning model based on global and local mutual information maximization. The supervised methods that take into account the labels from a specific downstream task to obtain path representations, which are: (1) **DeepGTT** [19] is a *supervised* travel time distribution estimation (i.e., to learn the parameters for inverse Gaussian distribution) framework based on a deep generative model. (2) **HMTRL** [46] enables unified route representation learning and exploits both spatio-temporal dependencies in road networks and the semantic coherence of historical routes. (3) **PathRank** [20] is a *supervised* path representation learning model based on GRUs. (4) **GCN** [49] is a graph covolutional neural network based method that estimates the travel times of all edges in a road network. The travel time of a path is then the sum of the travel times of the edges in the path. (5) **STGCN** [50] is a traffic prediction framework based on spatio-temporal graph convolutional networks. Similar to GCN, the travel time of a path is the sum of the predicted travel times of the edges in the path. Finally, note that for the path ranking task, we cannot simply aggregate the rankings of edges to obtain the rankings of paths. This also applies to the newly included path recommendation task. Thus, GCNs and STGCNs cannot work as baselines for the these two tasks.

4) Models for Downstream Tasks: For all unsupervised learning approaches, we first obtain a task-independent TPR and then apply a regression model to address different downstream tasks using task-specific labels. In the experiments, we use the ensemble model *Gradient Boosting Regressor* (*GBR*) to estimate travel time and ranking scores for paths as they are regression problems. In addition, we use the ensemble model *Gradient Boosting Classifier* (*GBC*) to make path recommendations, as they are classification problems.

TABLE I: Overall Accuracy on Travel Time Estimation

Methods	Aalborg		Harbin		Chengdu	
	MAE	MAPE	MAE	MAPE	MAE	MAPE
Node2vec	63.82	45.67	269.21	31.41	290.47	34.43
DGI	67.22	49.36	288.09	34.01	312.28	38.46
GMI	70.61	52.40	310.39	36.60	337.06	41.58
MB	57.32	39.37	315.25	35.28	333.73	42.45
BERT	71.96	45.42	217.96	24.52	303.00	36.77
InfoGraph	69.36	41.28	200.81	22.68	291.54	36.07
PIM	57.66	39.34	196.06	21.96	289.10	35.55
DeepGTT	44.78	26.53	214.95	22.76	305.08	35.47
HMTRL	40.59	21.81	228.58	23.60	360.08	37.33
PathRank	37.09	23.89	190.08	20.12	334.94	35.11
GCN	78.04	53.05	368.21	35.62	480.83	42.01
STGCN	58.57	38.97	284.12	23.48	406.09	33.58
WSCCL	**31.66**	**21.39**	**178.89**	**19.43**	**281.20**	**33.30**

5) Weak Labels: We consider two different types of weak labels, including peak/off-peak (POP) and traffic congestion indices (TCI). We use POP as default weak labels. We only conduct experiments on the Harbin and Chengdu data sets for TCI since we cannot obtain TCI for Aalborg, Denmark from Baidu Maps[3].

6) Implementation Settings: We set embedding feature dimensions of RT, NoL, OW, and TS as $d_{rt} = 64$, $d_l = 32$, $d_o = 16$, $d_{ts} = 16$, respectively. The feature dimensions of output from *Node2Vec* on both temporal graph and road networks are set to be 128. Meanwhile, we apply 2 LSTM layers and set the dimensionality of the hidden state h_j to 128. Further, we set the size for temporal path representation dimensionality to 128, i.e., $d_h = 128$. The number of Meta-Set is set to be $N = 10$, and the number of stages in curriculum learning is also set to be $M = 10$. The hyper-parameter λ is set to 0.8. We set the learning rate (lr) to $3e-4$ and the batch size 32. In particular, we train our *WSCCL* using all unlabeled paths shown in data sets section and then we randomly choose 80% and 20% paths in labeled path as training and testing data for GBR. Finally, we evaluate all models on a powerful Linux server with 40 Intel(R) Xeon(R) Gold 5215 CPUs @ 2.50GHz and four Quadro RTX 8000 GPU cards. Finally, all algorithm are implemented in PyTorch 1.9.1. The code is available at https://github.com/Sean-Bin-Yang/TPR.git and the CoRR version is available at https://arxiv.org/abs/2203.16110.

B. Experimental Results

*1) **Overall accuracy on downstream tasks**:* Table I, Table II and Table III report the overall results on the three downstream tasks. *WSCCL* achieves the best performance on these three tasks for three real-world data sets. The three graph node representation learning methods *Node2vec*, *DGI*, and *GMI* are unable to capture temporal correlation in the temporal path. In contrast, *WSCCL* takes temporal correlation into consideration by virtue of its temporal embedding layer. In addition, the weakly supervised contrastive curriculum learning improves the estimation accuracy.

[3]https://jiaotong.baidu.com/congestion/city/urbanrealtime/

TABLE II: Overall Accuracy on Path Rank Estimation

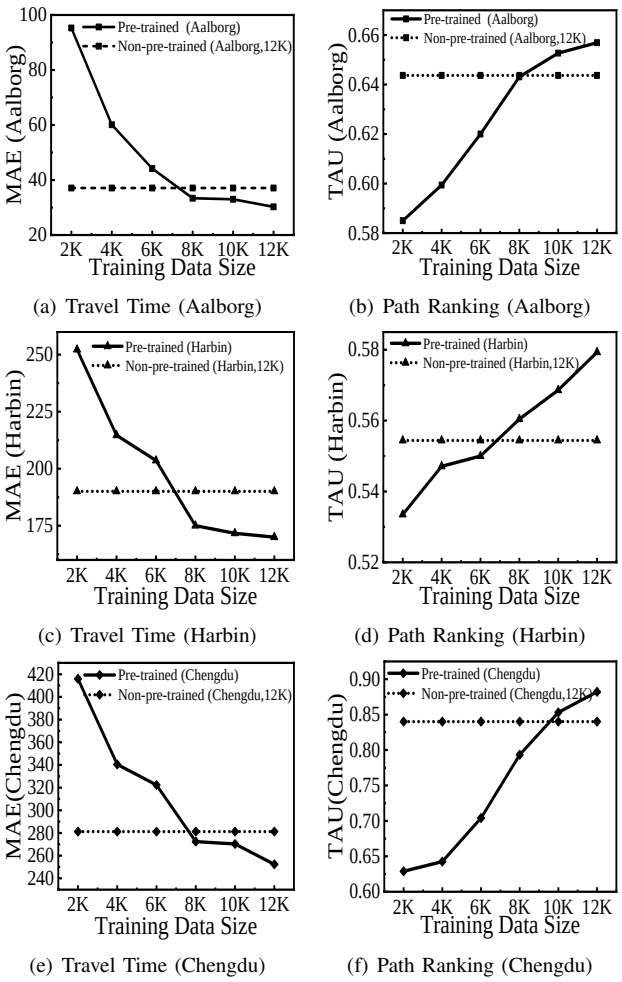
Methods	Aalborg		Harbin		Chengdu	
	MAE	τ	MAE	τ	MAE	τ
Node2vec	0.23	0.60	0.22	0.37	0.20	0.73
DGI	0.24	0.60	0.21	0.48	0.21	0.52
GMI	0.24	0.59	0.21	0.49	0.21	0.51
MB	0.23	0.62	0.22	0.44	0.20	0.71
BERT	0.26	0.49	0.22	0.46	0.22	0.55
InfoGraph	0.26	0.52	0.21	0.45	0.20	0.73
PIM	0.22	0.60	0.21	0.43	0.19	0.76
DeepGTT	0.39	0.12	0.29	0.04	0.23	0.20
HMTRL	0.17	0.65	0.22	0.51	0.16	0.77
PathRank	0.23	0.64	0.18	0.55	0.17	0.79
WSCCL	**0.15**	**0.68**	**0.14**	**0.68**	**0.13**	**0.84**

TABLE III: Overall Performance on Path Recommendation

Methods	Aalborg		Harbin		Chengdu	
	Acc.	HR	Acc.	HR	Acc.	HR
Node2vec	0.79	0.51	0.76	0.51	0.75	0.61
DGI	0.74	0.55	0.70	0.36	0.70	0.57
GMI	0.78	0.53	0.72	0.41	0.68	0.58
MB	0.67	0.48	0.61	0.69	0.73	0.69
BERT	0.60	0.43	0.64	0.53	0.66	0.61
InfoGraph	0.72	0.69	0.79	0.78	0.73	0.65
PIM	0.79	0.82	0.86	0.83	0.76	0.74
HMTRL	0.80	0.86	0.81	0.82	0.78	0.83
PathRank	0.77	0.71	0.79	0.74	0.77	0.73
WSCCL	**0.82**	**0.88**	**0.97**	**0.91**	**0.81**	**0.90**

Although *MB* and *BERT* can capture dependencies among the edge feature vectors in temporal paths, these approaches achieve poor estimation accuracy. This is because *MB* needs large amounts of negative samples to ensure effective training, which is not feasible in our scenario. In addition, *BERT* is not well suited for our setting of learning generic TPRs since BERT cannot support contrastive learning in the setting of multiple positive samples against multiple negative samples.

InfoGraph learns full graph representations. However, it only applies a local view and cannot capture the sequential information of edge in a path. In contrast, *WSCCL* not only uses sequence model (e.g., LSTM) to capture sequential information between edge in a path but considers both the (global) path and (local) edge levels. Although *PIM* is designed for path representation learning, it is unable to learn meaningful TPRs because it ignores temporal information and only has one positive sample. In contrast, *WSCCL* allows multiple positive temporal path samples in each minibatch and also uses a learned curriculum instead of the pre-defined curriculum negative sampling used by *PIM*. The supervised learning methods *DeepGTT*, *HMTRL*, and *PathRank* achieve relatively poor accuracy due to the small size of labeled training data. Since task-specific labels ("strong labels") are expensive to obtain, we consider a setting where labelled training data is limited. *DeepGTT* exhibits the worst performance on the Path Ranking task because it is designed for travel-time estimation, which is evidence for the poor generalizability of supervised feature representation learning, as discussed in Section I. In contrast, the *GCN* and *STGCN* results also are worse than the *WSCCL* results. This is because dependencies among edges are disregarded.



(a) Travel Time (Aalborg) (b) Path Ranking (Aalborg)

(c) Travel Time (Harbin) (d) Path Ranking (Harbin)

(e) Travel Time (Chengdu) (f) Path Ranking (Chengdu)

Fig. 7: Effects of Pre-training

*2) **Using* **WSCCL** *as a Pre-training Method:*** We conduct experiments that treat *WSCCL* as a pre-training method for the supervised method *PathRank*. Here, *PathRank* takes as input a sequence of edge features and estimates the travel time and ranking score. To use *WSCCL* for pre-training method for *PathRank*, we first train *WSCCL* in a weakly supervised manner and then apply the learned parameters in temporal path encoder to initialize the encoder in *PathRank*.

Fig. 7 shows performance of *PathRank* with and without pre-training for the two tasks. Without pre-training, *PathRank* is trained by using 12K labeled training paths. We observe the following: 1) With pre-training we can achieve the same performance as with non-pre-trained *PathRank* while using fewer labeled training paths. For example, when using *WSCCL* for pre-training, *PathRank* only needs 8K, 7K and 10K labeled samples for the Aalborg, Harbin and Chengdu data sets, respectively, to achieve the same performance as *PathRank* with 12K labeled samples, on the path ranking task. 2) When we pre-train *PathRank* with 12K samples, the performance is much better than without pre-training. In both tasks, we obtain similar observations, which indicates that *WSCCL* can

TABLE IV: Effect of the CL Design Strategy

Methods	Aalborg					
	Travel Time Estimation			Path Ranking		
	MAE	MARE	MAPE	MAE	τ	ρ
Heuristic	33.58	0.15	22.06	0.19	0.61	0.65
WSCCL	**31.66**	**0.13**	**21.39**	**0.15**	**0.68**	**0.72**

TABLE V: Effects of CL, Global Loss and Local Loss

Methods	Aalborg					
	Travel Time Estimation			Path Ranking		
	MAE	MARE	MAPE	MAE	τ	ρ
w/o CL	32.58	0.14	21.88	0.19	0.62	0.66
w/o Global	51.19	0.22	31.13	0.24	0.54	0.58
w/o Local	32.80	0.14	23.34	0.21	0.57	0.62
WSCCL	**31.66**	**0.14**	**21.39**	**0.15**	**0.68**	**0.72**

be applied advantageously for the pre-training of different downstream tasks.

*3) **Ablation Studies**:* We conduct ablation studies on *WSCCL* to observe 1) the effect of the CL design strategy on TPR learning; 2) the effect of variants of *WSCCL*, specifically *CL*, *global loss*, and local loss; 3) the effect of different weak labels; and 4) the effect of temporal information.

a) Effect of the CL Design Strategy: To observe the effectiveness of the learned CL, we compare with *WSCCL* with a heuristic curriculum design where we simply sort the paths based on the number of edges. The comparison between these two CL variants is reported in Table IV that shows that our learned curriculum is better than the heuristic curriculum design on all tasks over on Aalborg data sets. This is because the lengths of edges are varying (from few meters to kilometers), even two paths with the same number of edges, the lengths of two paths may have a big difference. Thus, the difficulties of paths cannot simply represented by the number of edges directly.

b) Effects of Global Loss, Local Loss, and CL: To study the effect of these three modules, we consider three variants of *WSCCL*: 1) *w/o Global*, 2) *w/o Local*, and 3) *w/o CL*. In *w/o Global*, we remove Global WSC loss from *WSCCL*, in *WSCCL w/o Local*, we remove the Local WSC loss, and in *WSCCL w/o CL*, the curriculum strategy is omitted. The results on Aalborg data sets are reported in Table V. We can observe that *WSCCL w/o Global* shows the worst performance and has a clear margin to the other variants. This shows that the proposed Global WSC performs well. We also observe that *WSCCL* achieves the best performance. This indicates that all the proposed modules contribute positively to the final performance, which validates the overall design.

c) Effect of Different Weak Labels: We conduct additional experiments by using traffic congestion indices (TCI), which indicate four congestion levels in a city across time, as weak labels. Table VI shows the results on the Harbin and Chengdu data sets. We observe that *WSCCL* works well when using the TCI as weak labels.

d) Effect of Temporal Information: We further conduct experiments on the three data sets using a *WSCCL* variant that disregards temporal information. The results, shown in

TABLE VI: Effect of Different Weak Labels

Methods	Harbin					
	Travel Time Estimation			Path Ranking		
	MAE	MARE	MAPE	MAE	τ	ρ
WSCCL-TCI	177.07	0.18	19.19	0.13	0.70	0.74
WSCCL-POP	178.89	0.18	19.43	0.14	0.68	0.73
Methods	**Chengdu**					
	Travel Time Estimation			**Path Ranking**		
	MAE	**MARE**	**MAPE**	**MAE**	τ	ρ
WSCCL-TCI	280.85	0.29	32.97	0.12	0.86	0.87
WSCCL-POP	281.20	0.29	33.30	0.13	0.84	0.86

TABLE VII: Effect of Temporal Information

Methods	Aalborg					
	Travel Time Estimation			PathRank		
	MAE	MARE	MAPE	MAE	τ	ρ
WSCCL	**31.66**	**0.14**	**21.39**	**0.15**	**0.68**	**0.72**
WSCCL-NT	41.25	0.18	29.38	0.21	0.55	0.59
Methods	**Harbin**					
	Travel Time Estimation			**PathRank**		
	MAE	**MARE**	**MAPE**	**MAE**	τ	ρ
WSCCL	**178.89**	**0.18**	**19.43**	**0.14**	**0.68**	**0.73**
WSCCL-NT	199.58	0.20	22.20	0.15	0.64	0.68
Methods	**Chengdu**					
	Travel Time Estimation			**PathRank**		
	MAE	**MARE**	**MAPE**	**MAE**	τ	ρ
WSCCL	**281.20**	**0.29**	**33.30**	**0.13**	**0.84**	**0.86**
WSCCL-NT	292.76	0.31	35.10	0.18	0.81	0.83

Table VII, indicate that the non-temporal *WSCCL-NT* performs worse than *WSCCL* on both downstream tasks, suggesting that our temporal embedding is effective.

*4) **Comparison with Temporally Enhanced Unsupervised Method**:* To compare *WSCCL* with the unsupervised *PIM* method, we first incorporate a temporal representation into the non-temporal path representations learned by *PIM*. Specifically, we use the same temporal embedding to learn temporal representations and then concatenate these with the path representation from PIM to obtain *PIM-Temporal* unsupervised TPRs. The results of comparing this approach with *WSCCL* are reported in Table VIII. We see that *WSCCL* outperforms *PIM-Temporal* on both tasks. This indicates that TPRs obtained by adding a temporal representation to the path representation directly is not as good as the TPR learned by *WSCCL*. This is because the added temporal representation can only capture the overall traffic condition on the road network for all the paths, yet not more unique spatio-temporal path representations learned by our *Temporal Path Encoder* for different paths. This experiment shows that it is not feasible to obtain a generic TRP by independently adding a temporal representation to an unsupervised learned generic PR (i.e., spatial representation). Further, it offers evidence of a more correlated and intricate interplay between space and time in TPRs. For example, during the morning peak hours, different paths may have different traffic conditions.

*5) **Comparison with Supervised Method**:* To study the applicability of TPRs from supervised method across tasks, we use the supervised methods *PathRank*, *HMTRL* and *DeepGTT*

TABLE VIII: Comparison with Temporally Enhanced Unsupervised *PIM* Method

Methods	Aalborg					
	Travel Time Estimation			Path Ranking		
	MAE	MARE	MAPE	MAE	τ	ρ
PIM-Temporal	42.27	0.19	27.95	0.19	0.65	0.70
WSCCL	**31.66**	**0.13**	**21.39**	**0.15**	**0.68**	**0.72**

TABLE IX: Comparison with Supervised Methods

Methods	Aalborg					
	Travel Time Estimation			Path Ranking		
	MAE	MARE	MAPE	MAE	τ	ρ
PathRank-PR	37.09	0.16	23.89	0.24	0.58	0.62
PathRank-TTE	55.08	0.24	36.71	0.23	0.64	0.68
HMTRL-PR	40.59	0.18	21.81	0.25	0.60	0.64
HMTRL-TTE	47.22	0.21	29.97	0.17	0.65	0.68
DeepGTT-PR	44.78	0.20	26.53	0.31	0.56	0.57
DeepGTT-TTE	59.52	0.26	37.80	0.39	0.12	0.12
WSCCL	**31.66**	**0.13**	**21.39**	**0.15**	**0.68**	**0.72**

as baselines. In the supervised methods, we define a primary and a secondary task: A supervised model is trained on the primary task, and then the learned path representation is applied to the secondary task directly. Thus, we have two experimental settings: 1) *Baseline-PR*, travel-time estimation is the primary task and path ranking is the secondary task; 2) *Baseline-TTE*, where path ranking is the primary task and travel-time estimation is the secondary task. The results are reported in Table IX. We first observe that *WSCCL* achieves the best performance on both downstream tasks. Further, we observe that the performance of *PathRank* and *HMTLR* are always better on the primary task than on the secondary task. For example, for travel-time estimation on Aalborg, *PathRank-PR* is better than *PathRank-TTE*. This is evidence of the drawbacks of supervised approaches that task-specific TPRs do not generalize well across tasks. Moreover, we also observe that *DeeGTT-PR* is always better than *DeeGTT-TTE* on both data sets. This is because *DeepGTT* is designed to do travel time distribution estimation. Given the task of path ranking, whose distribution may not follow the same inverse-Gaussian distribution like in travel time, so it fails in this case.

*6) **Parameter Studies**:* We study the effects of λ and N.

a) Effects of λ: To study the effect of the balancing factor λ (cf. Eq. 12), we conduct a parameter study on Aalborg. Based on the results reported in Table X, we see that the performance of our model changes when varying λ. We can also see that the optimal λ is 0.8, which means that both global WSC loss and local WSC loss can contribute to the model's performance. When $\lambda = 0.0$, the global WSC loss is ignored, which yields poor performance. When $\lambda = 1.0$, the local contrastive loss is ignored, and the best performance is not obtained, although the performance is quite good. When $\lambda > 0$, meaning that we consider both weakly-supervised and local WSC loss, we observe that the prediction performance is improved. Overall, we conclude that global WSC loss is more important than the local loss.

TABLE X: Effects of λ

λ	Aalborg					
	Travel Time Estimation			Path Ranking		
	MAE	MARE	MAPE	MAE	τ	ρ
0.0	51.19	0.22	31.13	0.24	0.54	0.58
0.2	40.25	0.18	24.67	0.22	0.60	0.64
0.4	34.22	0.15	21.80	0.18	0.64	0.68
0.6	34.76	0.15	22.35	0.17	0.65	0.69
0.8	**31.66**	**0.14**	**21.39**	**0.15**	**0.68**	**0.72**
1.0	32.80	0.14	23.34	0.21	0.57	0.62

TABLE XI: Effects of Number of Meta-Set.

N	Aalborg					
	Travel Time Estimation			Path Ranking		
	MAE	MARE	MAPE	MAE	τ	ρ
2	36.64	0.17	25.86	0.20	0.56	0.60
6	36.06	0.16	24.96	0.20	0.56	0.60
10	**31.66**	**0.14**	**21.39**	**0.15**	**0.68**	**0.72**
14	33.16	0.15	21.60	0.19	0.58	0.63
18	33.47	0.15	21.65	0.20	0.56	0.61

b) Effects of N: To study the effect of varying the number of *Experts* N, which is also the number of curriculum stages as we always set $N = M$, in the curriculum strategy, we observe the performance for values of N in $\{2, 6, 10, 14, 18\}$. The results in Table XI indicate that the best performance is obtained for $N = 10$ on Aalborg data set. We also observe that when N is too small, the curriculum strategy is not effective. This occurs because the number of *Experts* is small, meaning that the difficulty scores have more uncertainty and inaccuracy. This can also be the reason why the number of curriculum stages is small in the curriculum sample selection stage, such that the samples in the beginning are difficult to learn. Next, when the N becomes too large, this may cause problems in the curriculum sample selection stage because the diversity and number of data points in the meta-sets may be too small, resulting in the model overfitting at each stage.

VIII. Conclusions and Future Work

We study temporal path representation learning using weak labels. We propose a novel weakly supervised contrastive learning method that uses weakly supervised contrastive learning and local constrative loss. Next, we integrate curriculum learning into the method to further enhance its performance. Finally, we report on experiments on three data sets in the settings of three downstream tasks, finding that our proposal achieves significant performance improvements over unsupervised and supervised baselines. In addition, the proposed method can be utilized as a pre-training method to enhance supervised temporal path representation learning. As future work, it is of interest to study how to incorporate additional weak labels such as drivers and vehicle types.

Acknowledgments

This work was supported in part by Independent Research Fund Denmark under agreements 8022-00246B and 8048-00038B, the VILLUM FONDEN under agreements 34328

and 40567, Huawei Cloud Database Innovation Lab, and the Innovation Fund Denmark centre, DIREC.

REFERENCES

[1] C. Guo, B. Yang, J. Hu, C. S. Jensen, and L. Chen, "Context-aware, preference-based vehicle routing," *VLDB J.*, vol. 29, no. 5, pp. 1149–1170, 2020.

[2] J. Hu, B. Yang, C. Guo, and C. S. Jensen, "Risk-aware path selection with time-varying, uncertain travel costs: a time series approach," *VLDB J.*, vol. 27, no. 2, pp. 179–200, 2018.

[3] C. Guo, B. Yang, J. Hu, and C. S. Jensen, "Learning to route with sparse trajectory sets," in *ICDE*, 2018, pp. 1073–1084.

[4] S. A. Pedersen, B. Yang, and C. S. Jensen, "Anytime stochastic routing with hybrid learning," *Proc. VLDB Endow.*, vol. 13, no. 9, pp. 1555–1567, 2020.

[5] H. Liu, C. Jin, B. Yang, and A. Zhou, "Finding top-k optimal sequenced routes," in *ICDE*, 2018, pp. 569–580.

[6] S. A. Pedersen, B. Yang, and C. S. Jensen, "Fast stochastic routing under time-varying uncertainty," *VLDB J.*, vol. 29, no. 4, pp. 819–839, 2020.

[7] B. Zheng, Q. Hu, L. Ming, J. Hu, L. Chen, K. Zheng, and C. S. Jensen, "Soup: Spatial-temporal demand forecasting and competitive supply," *TKDE*, 2021.

[8] B. Yang, J. Dai, C. Guo, C. S. Jensen, and J. Hu, "PACE: a path-centric paradigm for stochastic path finding," *VLDB J.*, vol. 27, no. 2, pp. 153–178, 2018.

[9] J. Hu, B. Yang, C. Guo, C. S. Jensen, and H. Xiong, "Stochastic origin-destination matrix forecasting using dual-stage graph convolutional, recurrent neural networks," in *ICDE*, 2020, pp. 1417–1428.

[10] T. Kieu, B. Yang, C. Guo, and C. S. Jensen, "Distinguishing trajectories from different drivers using incompletely labeled trajectories," in *CIKM*, 2018, pp. 863–872.

[11] J. Hu, C. Guo, B. Yang, and C. S. Jensen, "Stochastic weight completion for road networks using graph convolutional networks," in *ICDE*, 2019, pp. 1274–1285.

[12] R. Cirstea, T. Kieu, C. Guo, B. Yang, and S. J. Pan, "EnhanceNet: Plugin neural networks for enhancing correlated time series forecasting," in *ICDE*, 2021, pp. 1739–1750.

[13] R. Cirstea, B. Yang, C. Guo, T. Kieu, and S. Pan, "Towards spatio-temporal aware traffic time series forecasting," in *ICDE*, 2022.

[14] X. Wu, D. Zhang, C. Guo, C. He, B. Yang, and C. S. Jensen, "AutoCTS: Automated correlated time series forecasting," *Proc. VLDB Endow.*, vol. 15, no. 4, pp. 971–983, 2022.

[15] D. Campos, T. Kieu, C. Guo, F. Huang, K. Zheng, B. Yang, and C. S. Jensen, "Unsupervised time series outlier detection with diversity-driven convolutional ensembles," *Proc. VLDB Endow.*, vol. 15, no. 3, pp. 611–623, 2021.

[16] T. Kieu, B. Yang, C. Guo, R. Cirstea, Y. Zhao, Y. Song, and C. S. Jensen, "Anomaly detection in time series with robust variational quasi-recurrent autoencoders," in *ICDE*, 2022.

[17] T. Kieu, B. Yang, C. Guo, C. S. Jensen, Y. Zhao, F. Huang, and K. Zheng, "Robust and explainable autoencoders for time series outlier detection," in *ICDE*, 2022.

[18] S. B. Yang, C. Guo, J. Hu, J. Tang, and B. Yang, "Unsupervised path representation learning with curriculum negative sampling," in *IJCAI*, 2021, pp. 3286–3292.

[19] X. Li, G. Cong, A. Sun, and Y. Cheng, "Learning travel time distributions with deep generative model," in *WWW*, 2019, pp. 1017–1027.

[20] S. B. Yang, C. Guo, and B. Yang, "Context-aware path ranking in road networks," *TKDE (Early Access)*, 2020.

[21] J. Li, Z. Han, H. Cheng, J. Su, P. Wang, J. Zhang, and L. Pan, "Predicting path failure in time-evolving graphs," in *KDD*, 2019, pp. 1279–1289.

[22] F. Sun, J. Hoffmann, V. Verma, and J. Tang, "Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization," in *ICLR*, 2020.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[24] J. Chung, Ç. Gülçehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *CoRR*, vol. abs/1412.3555, 2014.

[25] C. Li, J. Ma, X. Guo, and Q. Mei, "Deepcas: An end-to-end predictor of information cascades," in *WWW*, 2017, pp. 577–586.

[26] Z. Liu, V. W. Zheng, Z. Zhao, F. Zhu, K. C. Chang, M. Wu, and J. Ying, "Semantic proximity search on heterogeneous graph by proximity embedding," in *AAAI*, 2017, pp. 154–160.

[27] R. D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, and Y. Bengio, "Learning deep representations by mutual information estimation and maximization," in *ICLR*, 2019.

[28] P. Velickovic, W. Fedus, W. L. Hamilton, P. Liò, Y. Bengio, and R. D. Hjelm, "Deep graph infomax," in *ICLR*, 2019.

[29] Z. Peng, W. Huang, M. Luo, Q. Zheng, Y. Rong, T. Xu, and J. Huang, "Graph representation learning via graphical mutual information maximization," in *WWW*, 2020, pp. 259–270.

[30] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, and D. Krishnan, "Supervised contrastive learning," in *NeurIPS*, 2020, pp. 18 661–18 673.

[31] J. Bromley, I. Guyon, Y. LeCun, E. Säckinger, and R. Shah, "Signature verification using a siamese time delay neural network," in *NIPS*. Morgan Kaufmann, 1993, pp. 737–744.

[32] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *ICML*, vol. 382, 2009, pp. 41–48.

[33] Y. Huang, Y. Wang, Y. Tai, X. Liu, P. Shen, S. Li, J. Li, and F. Huang, "Curricularface: Adaptive curriculum learning loss for deep face recognition," in *CVPR*, 2020, pp. 5900–5909.

[34] Y. Kong, L. Liu, J. Wang, and D. Tao, "Adaptive curriculum learning," in *ICCV*, 2021, pp. 5067–5076.

[35] Y. Wang, W. Gan, J. Yang, W. Wu, and J. Yan, "Dynamic curriculum learning for imbalanced data classification," in *ICCV*, 2019, pp. 5016–5025.

[36] L. Shen and Y. Feng, "CDL: curriculum dual learning for emotion-controllable response generation," in *ACL*, 2020, pp. 556–566.

[37] C. Wang, Y. Wu, S. Liu, M. Zhou, and Z. Yang, "Curriculum pre-training for end-to-end speech translation," in *ACL*, 2020, pp. 3728–3738.

[38] B. Xu, L. Zhang, Z. Mao, Q. Wang, H. Xie, and Y. Zhang, "Curriculum learning for natural language understanding," in *ACL*, 2020, pp. 6095–6104.

[39] H. Yuan, G. Li, Z. Bao, and L. Feng, "Effective travel time estimation: When historical trajectories over road networks matter," in *SIGMOD*, 2020, pp. 2135–2149.

[40] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *KDD*, 2016, pp. 855–864.

[41] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[42] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017, pp. 5998–6008.

[43] P. Newson and J. Krumm, "Hidden markov map matching through noise and sparseness," in *SIGSPATIAL*, 2009, pp. 336–343.

[44] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.

[45] J. H. Zar, "Significance testing of the spearman rank correlation coefficient," *Journal of the American Statistical Association*, vol. 67, no. 339, pp. 578–580, 1972.

[46] H. Liu, J. Han, Y. Fu, J. Zhou, X. Lu, and H. Xiong, "Multi-modal transportation recommendation with unified route representation learning," *Proc. VLDB Endow.*, vol. 14, no. 3, pp. 342–350, 2020.

[47] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin, "Unsupervised feature learning via non-parametric instance discrimination," in *CVPR*, 2018, pp. 3733–3742.

[48] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *NAACL-HLT*, 2019, pp. 4171–4186.

[49] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *NIPS*, 2016, pp. 3837–3845.

[50] B. Yu, H. Yin, and Z. Zhu, "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting," in *IJCAI*, 2018, pp. 3634–3640.