Linking Mobility Traces of the Same User Across Different Datasets

Dylan Zhang
School of Physical and Mathematical Sciences
Nanyang Technological University
Singapore
M180175@e.ntu.edu.sg

Homin Park*
School of Science and Technology
Singapore University of Social Sciences
Singapore
hominpark@suss.edu.sg

Shili Xiang
Institute for Infocomm Research
*A*STAR*
Singapore
sxiang@i2r.a-star.edu.sg

***Abstract*—Smart mobile apps have become an integral part of our daily lives, offering intelligent services anytime, anywhere. Various apps tap into mobility traces from different aspects of our daily activities, each managing these traces independently. Linking these isolated traces from different apps offers significant potential for gaining a deeper understanding of user behavior. Various solutions have been proposed to connect mobility traces of the same user across different apps, showing effectiveness in certain scenarios. However, we believe they have not effectively tackled the following four key challenges. First, a solution must accommodate a growing number of users. Second, mobility traces of newly joined users must be successfully linked. Third, auxiliary information, such as place of interests, might not always be accessible. Fourth, activity traces could be extremely sparse. In this paper, we introduce TLink, a deep learning framework designed to link mobility traces of the same user across different datasets, to address the above challenges. TLink employs a temporal convolutional network as its encoder and assesses the probability that a pair of traces, sourced from different datasets, originates from the same user. To train the model effectively, we adapted a multi-step training procedure that strategically modulates the complexity of the training objective. Furthermore, we introduce a new metric learning objective specifically crafted for the trace linking task. Our evaluation demonstrates a notable performance of TLink under challenging scenarios, evidencing a performance improvement of more than tenfold compared to an applicable baseline.**

***Index Terms*—Spatio-Temporal Data, Metric Learning, Deep Learning**

I. Introduction

The ubiquity of smart mobile devices and apps in our modern lives is undeniable. We enjoy intelligent services anywhere and anytime, which allows these apps to collect vast amounts of spatio-temporal mobility traces, composed of location and timestamps, from our daily activities. However, each app typically captures distinct segments of our day and manages them independently. For instance, while we might hail a ride using apps like Uber or Grab, we might later make payments with digital wallets such as Samsung Pay or Apple Pay. Merging these isolated traces of user behaviors from different apps can lead to a profound and comprehensive understanding of users. Such insights not only bolster public safety initiatives, like tracking the spread of infectious diseases, but also enhance user-centric offerings, such as smart apps with advanced recommendation systems.

Prior methods for linking mobility traces have primarily utilized classification-based approaches, operating under the assumption that users exhibit routine and predictable behaviors [1]–[5]. In these models, each class corresponds to a unique user. The objective is to associate a trace with the user whose behavior most closely aligns with the trace in question. However, these models come with several limitations. First, they depend on a pre-defined set of classes, requiring retraining when new users are introduced to the dataset. Second, newcomers must accumulate a significant volume of data before they can be effectively integrated. Third, models that utilize a classifier can handle only up to about one thousand users, even though the user count for an app can easily surge into the thousands or even millions. Lastly, these models suffer with accommodating users who exhibit unpredictable movement patterns, such as taxi drivers or delivery riders, whose routes are dictated primarily by on-demand needs.

Besides the challenges posed by classification-based approaches, we identified two additional problems. First, several existing methods rely on auxiliary information, such as location semantics [1]–[6]. While this information can enhance linking performance, it might not always be readily accessible. Second, many of these methods hinge on relatively dense mobility traces. However, in practice, some apps, like digital wallets, produce extremely sparse mobility traces due to their infrequent usage and brief duration of activity.

To address the limitations of existing approaches, we present TLink, a deep learning framework designed for linking mobility traces of the same user across different datasets. Rather than associating mobility traces to specific owners or measuring similarity between two traces, TLink assesses the binary likelihood that a pair of traces, sourced from different datasets, belong to the same user. TLink further distinguishes itself in three key areas: its architecture, loss function, and training procedure.

Concerning architecture, TLink is designed to leverage: 1)

This work is supported by the National Research Foundation, Prime Ministers Office, Singapore, under NRF-NSFC Joint Research Grant Call on Data Science [grant number NRF2016NRF-NSFC001-113].

The research work was conducted during the time the first and second authors were affiliated with the Institute for Infocomm Research, A*STAR, serving as an internship student and a research scientist, respectively.

*Corresponding author.

raw spatio-temporal data that consists solely of geographic coordinates and timestamps, devoid of auxiliary information; 2) a temporal convolutional network (TCN), specifically 1D-ResNet [7], as its encoder; and 3) a sigmoid function to measure the binary likelihood that two traces, sourced from different datasets, originate from the same user.

Regarding its loss function, TLink employs a metric learning technique. Among various state-of-the-art solutions, we explored the feasibility of a proxy-based approach, namely proxy anchor loss (PAL) [8]. However, our in-depth analyses shown in Section IV-B2 unveiled its limitations within the distinct context of mobility trace linking. To address this, we introduce binary proxy anchor loss (BPL), an adaptation of PAL specifically devised for trace linking.

Finally, TLink incorporates a multi-step training procedure. Distinguishing traces from the same user versus those from others is a nuanced undertaking. Challenges emerge when traces of the same user are scattered spatio-temporally, or when different users with analogous routines generate similar traces. This is further complicated by class imbalance problem where we have only one pair of traces from a specific user midst of thousands of pairs from others. To navigate these intricacies, we developed a multi-step training method where the complexity of the training objectives is strategically controlled, from easy to hard.

We evaluated TLink using two real-world spatio-temporal datasets, namely *trip* and *log*, collected by a major Singapore taxi company in August 2008. Our results demonstrate outstanding performance of TLink against existing methods. For example, DPLink-Lite, a modification of DPLink [6], achieved hit@1 of 0.04 and hit@20 of 0.15, whereas TLink scored hit@1 of 0.50 and hit@20 of 0.84. Additionally, our experiments highlighted the favorable impact of our architectural decisions, including the adoption of TCN, BPL, and other control parameters, as well as the efficacy of the multi-step training process.

II. Related Work

A. Human Mobility Trace Linking via Deep Learning

There are a number of mobility trace linking solutions exploiting deep learning classifiers [1]–[6]. For example, a series of Trajectory-User Linking (TUL) papers were initiated by TULER proposing a RNN-based classification model to learn the mapping from an input trace to its owner [1]. To address the data sparsity problem, TULVAE introduced the use of a semi-supervised Variational Auto-Encoder (VAE) leveraging unlabeled data [5]. DeepTUL adopted an attentive recurrent neural network, attempting to capture mobility patterns from historical traces [3]. However, such classification-based solutions have the following three major drawbacks.

Firstly, they struggle to accommodate a growing number of users due to several challenges and concerns. As the number of classes (users) increases, the final softmax computation, which calculates the probabilities for each class, becomes more computationally intensive and is prone to overfitting. Additionally, as the number of classes rises, the margin between these classes can shrink. This makes optimization more challenging due to more nuanced differences between each class.

Secondly, integrating newcomers can be time-consuming. Classifiers are traditionally designed to differentiate a fixed set of classes. When a new user is introduced, the classifier must be retrained. This retraining requires the newcomer to accumulate a significant amount of data first, to prevent a class imbalance problem where the classifier becomes biased towards classes with more examples.

Lastly, these classifiers suffer to accommodate users who display unpredictable movement patterns. To define a user, the model attempts to extract and identify their inherent behavioral patterns. However, certain users, like taxi drivers or delivery riders, often lack consistent mobility patterns, as their routes are highly influenced by on-demand needs.

To address the challenges posed by classification-based solutions, DPLink introduced a framework comprising a feature extractor and a comparator [6]. Instead of classifying mobility traces to each user, DPLink focused on scoring the similarity between two mobility traces. However, its performance hinges on three key factors: 1) the presence of users' historical records, 2) the availability of detailed location semantics, and 3) the presumption that users maintain consistent mobility patterns across different days.

B. Metric Learning

While classification predicts a discrete label for a given input by mapping it directly to predefined classes, metric learning focuses on discerning a similarity measure between data points. Given that this approach aligns well with TLink's needs, we explored the feasibility of various metric learning algorithms.

Metric learning methods can be broadly segmented into two categories: pair-based [9]–[11] and proxy-based losses [8], [12], [13]. Pair-based losses, such as contrastive loss [9], triplet loss [10], and N-pair loss [11], leverage data-to-data relations, aiming to draw similar samples closer and push dissimilar ones apart. Although these pair-based techniques capture nuanced semantics between data, they might face slow convergence and may require intricate sampling strategies to circumvent detrimental training data pairs [14].

On the other hand, proxy-based losses use class representatives to leverage data-to-proxy relations, often resulting in more robust outcomes compared to the pair-based losses. These methods typically converge faster and do not necessitate complex data pair sampling strategies [8]. An early work in this domain, Proxy-NCA, assigns a single proxy for each class, ensuring data and proxy pairs from the same class are brought closer while those from different classes are distanced [12]. The SoftTriple loss refines this concept by allocating multiple proxies for each class, addressing intra-class variance and enhancing class representation [13].

In recent years, the Proxy-Anchor Loss (PAL) has been introduced, harnessing the strengths of both pair and proxy-based methods while addressing their drawbacks [8]. By leveraging proxies, PAL accelerates convergence and exhibits

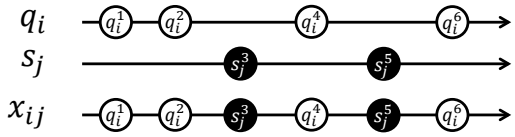


Fig. 1: An illustration of a merged trace x_{ij}.

resilience to noisy labels and outliers. Concurrently, it facilitates interactions between data points, exploiting data-to-data relationships. While PAL is regarded as a state-of-the-art method, we observed that, in its default configuration, it does not align well with the demands of mobility trace linking. Specifically, PAL was designed to tackle multi-class problems. Proxies are initialized randomly following a Gaussian distribution, and their positions are adjusted every training epoch.

III. TLink: Architecture and Method

A. Problem Formulation

Let $Q = \{q_1, ..., q_i, ..., q_{|Q|}\}$ and $S = \{s_1, ..., s_j, ..., s_{|S|}\}$ denote two different datasets, where q_i and s_j represent i^{th} and j^{th} users' mobility traces from Q and S, respectively. The total number of users is denoted as $|Q|$ and $|S|$, and Q and S shares the same set of users. Each mobility trace comprises of longitude, latitude, and timestamp in chronological order forming a spatio-temporal trace of a user. We merge q_i and s_j to generate x_{ij} where records from both traces are sorted in chronological order. As shown in Figure 1, where superscripts indicate the timestamps, if $q_i = \{q_i^1, q_i^2, q_i^4\}$ and $s_j = \{s_j^3, s_j^5\}$, then the merged sequence $x_{ij} = \{q_i^1, q_i^2, s_j^3, q_i^4, s_j^5\}$.

In practice, as depicted in Figure 2, for a given query trace q_i, TLink produces $|S|$ merged traces x_{ij}, which comprise one positive pair (from the same user) and $|S|-1$ negative pairs (from other users). TLink then evaluates the binary likelihood that each x_{ij} belongs to the same user and selects the top-k candidates.

B. Model Architecture

While most previous trace linking methods employ recurrent neural networks (RNNs) and their variations, we believe that they may suffer from 1) the vanishing gradient problem caused by the lengthy mobility traces, and 2) the expensive computational cost. To address such problems, we employ one of the temporal convolutional networks (TCN), namely the 1D residual network (1D-ResNet), as a backbone encoder [7].

As shown in Figure 3c, the final architecture consists of a 1D-ResNet encoder (two B1 blocks and one B2 block), a global average pooling layer, and two fully connected (FC) layers. The sigmoid function is attached to the output layer to measure the binary likelihood score $y(x_{ij})$.

The architectural details are as follows. Block 1 (B1) of the 1D-ResNet comprises three filters and one identity shortcut, with kernel sizes of 8, 5, 3, and 1, respectively. Block 2 (B2) contains three filters with the same configuration as Block 1 but omits the identity shortcut. The filter size of the initial B1, as illustrated in Figure 3c, is 32, the subsequent B1 is

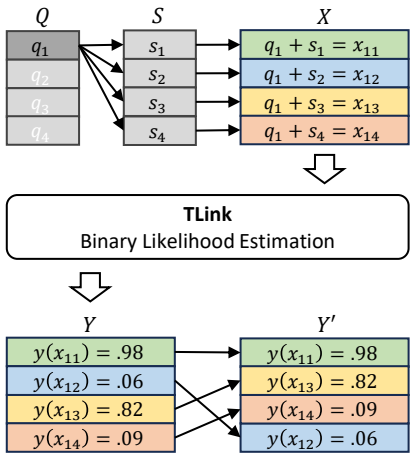
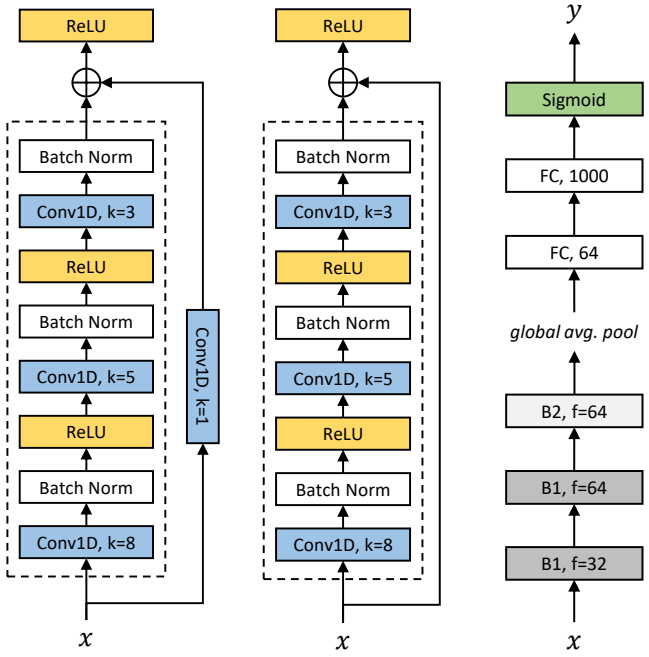


Fig. 2: An overview of TLink pipeline. For each merged trace x_{ij}, TLink model derives binary likelihood estimation (BLE) scores $y(x_{ij})$. These scores are then utilized to select the top-k candidates most likely corresponding to the same user. Y represents the collection of the BLE scores and Y' indicates the sorted version of Y.



(a) Block 1, B1 (b) Block 2, B2 (c) Architecture

Fig. 3: The architecture of TLink including 1D-ResNet, global average pooling, fully connected layers, and sigmoid function.

64, and B2 is 64. After global average pooling, the first FC layer comprises 64 nodes, while the second FC layer has 1,000 nodes.

C. Binary Proxy Anchor Loss

Proxy Anchor Loss (PAL) is one of the state-of-the-art methods harnessing the benefits of both pair and proxy-based

(a) BPL (b) Max-BPL (c) PAL (d) Max-PAL

Fig. 4: An illustration of BPL and PAL with different proxy and margin configurations. The proxies (colored stars) for BPL are oppositely initialized while PAL places them ranodmly. The margin between two classes are represented using the white and grey spaces.

approaches. However, after a comprehensive analysis, we noted that PAL, in its default configuration, is not suitable for binary likelihood estimation due to the following two reasons.

Firstly, PAL randomly initializes proxies using a Gaussian distribution to guarantee sufficient separation among multiple classes. However, since TLink is exclusively concerned with two classes, traces from the same user and those from others, it is essential to initialize the two proxies in direct opposition to ensure maximum separation between these classes.

Secondly, in PAL, proxies are updated through gradients, altering the distance between proxies with each epoch. However, TLink aims to attain optimal binary likelihood estimation performance, requiring maximal separation between two proxies once initialized oppositely. This consistency also allows us to have a stable representation of the two classes throughout multi-step training procedure where data distributions and characteristics drastically differ.

Therefore, we propose a binary proxy anchor loss (BPL), inspired by PAL, which employs two proxies oppositely initialized as shown in Figure 4b. The BPL can be expressed as follows:

$$l(X) = \frac{1}{|P_X^+|} \sum_{p \in P_X^+} log\left(1 + \sum_{x \in X_p^+} e^{-\alpha(\sigma(x,p)-\delta)}\right) + \frac{1}{|P|} \sum_{p \in P} log\left(1 + \sum_{x \in X_p^-} e^{-\alpha(\sigma(x,p)+\delta)}\right) \quad (1)$$

where X is the set of vectors in a batch, $\sigma(.,.)$ is the cosine similarity between two vectors, $\delta > 0$ is a margin, $\alpha > 0$ is a scaling factor, P is the set of all proxies, $|P|$ is two since we are dealing with binary problem, P_X^+ is the set of positive proxies of data in X, X_p^+ is the set of positive vectors of p, and X_p^- is the set of negative vectors of p acquired by $X - X_p^-$.

The margin δ is one of the important control parameters determining how the model will be trained. As shown in Figure 4a, if we decrease the margin, reducing the white area, the chance of having an overfitting problem increases since the model must squeeze all data points inside the white space. On the other hand, increasing the margin causes the boundary between two classes diminishes. Note that, to avoid an overlap of margins, we set $\delta \leq \sigma(p_{pos}, p_{neg})/2$, where p_{pos} is the proxy for the positive class and p_{neg} is for the negative. This is well illustrated in Figure 4b and 4d where the margin is set to the max.

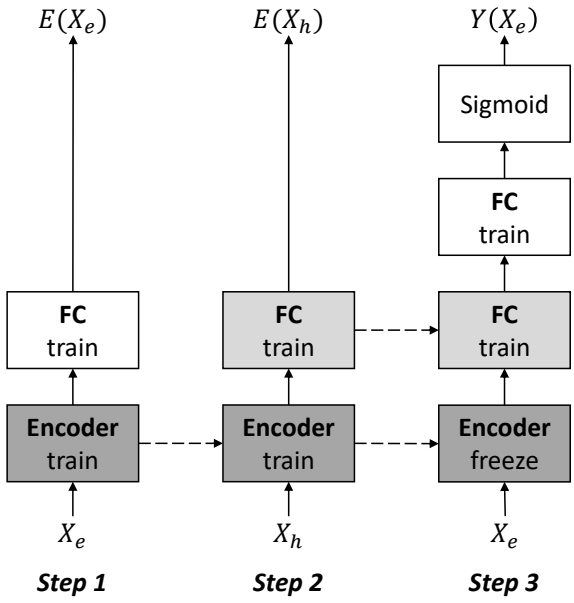


Fig. 5: An overview of the multi-step training procedure and varying architectures.

D. Multi-step Training Procedure

Differentiating the traces of the same user from others is a challenging task. For instance, two traces q_i and s_j of the same user could be spatio-temporally distant while another pair could exhibit a similar pattern but come from different users. This task is further complicated in practice by the class imbalance problem. As depicted in Figure 2, for a query trace q_i, TLink will generated only one merged trace x_{ij} of the same user compared to $|S|-1$ merged traces of the others.

To address these challenges, we propose a three-step training procedure in which the objectives and difficulties of each step are incrementally adjusted. By guiding TLink model to handle increasingly complex scenarios, the resultant model becomes adept at accurately assessing the binary likelihood scores of merged traces that originate from the same user. Details of each step is as follows.

1) Step 1: Learning the binary concept: The first step is to warm up the model to learn the difference between the positive (traces of the same user) and negative classes (traces of the others) in an ideal settings. For this, we curated an easy dataset X_e where the two classes are well-balanced and relatively easier to be differentiated. The easy dataset X_e is curated as follows.

First, we merge all traces from $|Q|$ and $|S|$ generating $|Q|\times|S|$ merged traces in total. Second, take the traces of the same user (positives) x_{ij} where $i = j$, giving us $|S|$ positive traces. Third, randomly select $|S|$ traces of the others

(negatives) x_{ij} where $i \neq j$. We sampled an equal number of positives and negatives to ensure a balance between the two classes.

Using this balanced dataset, we train the encoder (composed of two B1, one B2, and global average pooling layer as shown in Figure 3c) and the first FC layer with 64 nodes exploiting BPL. After converging, the model, shown in Step 1 of Figure 5, will generate embedded representations $E(X_e)$ that are clustered around two opposing proxies enabling relatively easier separation between the positive and negative classes.

2) Step 2: Adapting to the real-world data: Once the model learned about the binary concept through Step 1, we then train the model to learn about the subtle difference between two classes using the real-world dataset denoted as X_h. The dataset X_h includes $|Q|$ positives and $|Q|\times|S|-|Q|$ negative entailing all the challenges stated previously. To transfer the knowledge attained from Step 1, we 1) migrate the encoder and allow it to be finetuned, and 2) reset the first FC layer to be retrained. With such an architecture, also shown in Step 2 of Figure 5, we exploit BPL to train the model to form clusters around two proxies.

3) Step 3: Measuring the binary likelihood score: In the final step, the model is trained to estimate the binary likelihood score of the merged traces $y(x_{ij})$. The simplest approach is to exploit the distance between data points and their corresponding proxies. However, such a distance-based method proves ineffective, as proxies are designed primarily as anchors rather than accurate class representatives. They actively pull and push data points, clustering them within designated white spaces illustrated in Figure 4. In fact, once a data point is drawn into the appropriate white space, its loss value becomes zero regardless of its proximity to the proxy, requiring no further updates.

To effectively measure the binary likelihood score, we harness the model trained from Steps 1 and 2, which encompasses the encoder and the first FC layer. Building upon this foundation, we append an additional FC layer comprising 1,000 nodes and finalize the model with a sigmoid function at the tail end as shown in Step 3 of Figure 5. To protect the foundational model, the encoder and the first FC layer are frozen, preventing any subsequent modifications. Contrarily, the newly incorporated layers, the second FC layer and the sigmoid function, are comprehensive trained employing binary cross entropy loss. The Step 3 training is conducted using the easy dataset X_e.

IV. Evaluation

A. Experimental Setup

1) Dataset: Mobility trace linking problems require the use of two or more distinct real-world spatio-temporal datasets sharing a common identifiable user base. The details used to identify each user (e.g., username, residential information, and more) could raise significant privacy concerns when they are released to the public. In fact, most datasets are typically managed privately by research teams and their partners, a practice that helps mitigate privacy risks but can pose challenges for data accessibility and collaborative research. The authors of DPLink also state that ISP-Weibo dataset, the main dataset used in DPLink, is private data collected and processed by the research team and partners, which cannot be published to the public due to privacy concerns. Another dataset called Foursquare-Twitter, used by DPLink and introduced by Zhang et. al. [15], is also not open to the public.

To evaluate our solution, we employ two distinct spatio-temporal mobility trace datasets: *trip* and *log*. These datasets were collected by a major Singapore-based taxi company in August 2008. The *trip* dataset is composed of records manually initiated by taxi drivers whenever they activate or deactivate the taximeter, marking the start and end of a trip. Each of these records incorporates the taxi ID, start time, start location, end time, and end location. Conversely, the *log* dataset comprises records captured automatically at regular time intervals, detailing the taxi's location. A log record includes the taxi ID, its location, and the corresponding timestamp. Each user has one *trip* and one *log* record, thus the total number of records found in each dataset equals the total number of users $|S|$ and $|Q|$.

For training and testing the model, we randomly divided the users into two disjoint sets. The easy dataset X_e contains 400 users from the first disjoint set, while the hard dataset X_h contains 200 users. Specifically, X_e includes 400 positive examples and 400 negative examples, whereas X_h comprises 200 positives and 39,800 negatives. We conduct the testing of our model using a test dataset X_t that includes 1,000 random users selected from the second disjoint set. Lastly, we downsampled the traces to 100 data points, averaging 3.33 records per day, to challenge our model with extremely sparse data.

2) Baselines: In our work, we consider two categories of baselines: mobility trace linking solutions and metric learning losses. Firstly, most of the prior mobility trace linking solutions, such as TULER [1], TULVAE [5], and DeepTUL [3], cannot be directly compared with TLink. The prior works employ classification-based approaches which require a fixed set of users during training and testing. In fact, such methods cannot support the unseen users. DPLink, on the other hand, is a mobility trace linking solution that addresses challenges similar to those of TLink. However, DPLink leverages the presence of daily routines, historical records of each user, and rich location semantics. For a fair comparison, we removed modules associated with routine and location semantic analysis, resulting in a lighter version of DPLink (DPLink-Lite). For our evaluation, we set the hidden size of DPLink-Lite to 128. Secondly, to demonstrate the advantages of the binary proxy anchor loss (BPL), we employed three metric learning losses, including proxy-NCA [12], SoftTriple [13] and PAL [8].

3) Training Details: We set the learning rate as 0.001, embedding size as 64, and batch size as 64. The number of training epochs for each step is configured as follows: 2,000 epochs for Step 1, 500 epochs for Step 2, and 1,000 epochs for Step 3. For each experiment, we train five different model

instances and report their average performances.

B. Result Analysis

The feasibility of TLink is evaluated by examining the following three aspects: 1) the benefit of employing the multi-step training procedure, 2) the advantage of using BPL over other metric learning objectives, and 3) the performance of different binary likelihood similarity measures.

As shown in Table 1, we tested six different model configurations. For the encoder, we evaluated both TCN and GRU. Regarding the loss function, we examined ProxyNCA, SoftTriple, PAL, and BPL. Additionally, we assessed the impact of different proxy configurations, specifically random and opposite initialization, as well as the influence of updating proxies through gradient. Lastly, we experimented with two margin configurations: random and maximum margin value.

The performance of each model configuration is measured using hit@k, which measures the ratio of the merged trace x_{ij} of the same user listed in the top-k candidates.

1) The benefit of multi-step training procedure: As discussed in Section 3.D, differentiating the traces of the same user from those of others presents an immense challenge. Initially, we investigate the outcome of cold starting the model via Step 2, using X_h. Table 1 reveals that the performance of models trained under Step 2 & 3 significantly lags behind other models and configurations. Specifically, there is an approximate 0.373 difference in hit@1 between a model (using TCN and BPL, as shown in the last column of Table 1) trained through Step 2 & 3 and another trained through Step 1, 2, & 3.

To effectively warm up the model, TLink employs Step 1 with X_e. As indicated in the last column of Table 1, hit@1 for Step 1 & 3 stands at 0.416, whereas for Step 2 & 3, it is only 0.129, a difference of approximately 0.287. When it comes to hit@20, Step 2 & 3 achieves about 0.583, while Step 1 & 3 reaches around 0.821. Given that the test dataset X_t consists of 1,000 users, such a performance is indeed commendable.

While Step 1 & 3 already achieved an outstanding results, there is a room for further improvements. By integrating Step 1, 2, & 3 together, we observe an approximate gain of 0.1 on hit@1 (from 0.416 for Step 1 & 3 to 0.502 for Step 1, 2, & 3) and 0.02 on hit@20 (from 0.821 for Step 1 & 3 to 0.843 for Step 1, 2, & 3). The nuanced adaptation in Step 2 empowers the model to discern more finely between the two classes. This consistent performance boost across all model configurations underscores the efficacy of the multi-step training approach.

2) The advantage of using BPL over the other metric learning objectives: To evaluate the advantages of BPL, we employ different loss functions including proxyNCA [12], SoftTriple [13], and PAL [8]. For ProxyNCA, shown in the third column of Table 1, we randomly initialize the proxies and allow them to be updated through gradients while the margin is randomly configured. For SoftTriple, shown in the fourth column, three proxies per class are randomly initialized and updated through gradients while the margin is randomly configured. For PAL, shown in the fifth column, in its original

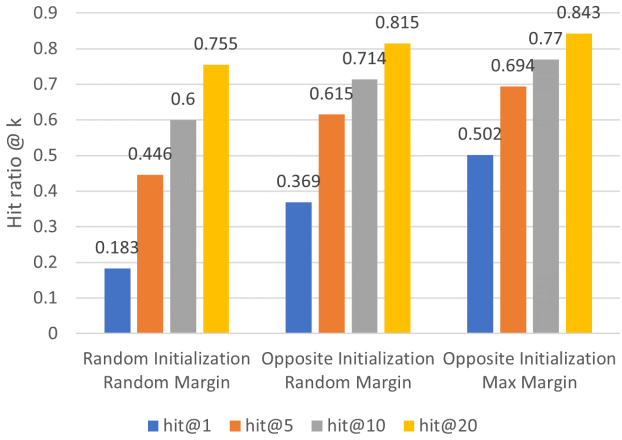


Fig. 6: Performance of BPL under different proxy and margin configurations.

form, proxies are randomly initialized and updated through gradients while the margin value is randomly determined. To test the impact of the proxy update through gradients, as shown in the sixth column, we disabled the update mechanism for PAL while keeping other settings the same. The binary proxy loss (BPL), shown in the last column of Table 1, initializes the proxies oppositely and disabled the update. For ease of evaluation, our discussions are based on the models using TCN and trained under Step 1, 2 & 3. Please note that similar performance differences are found for other settings as well.

Based on the numbers shown in Table 1, ProxyNCA and SoftTriple perform quite similarly. On the other hand, the original PAL performs slightly better than ProxyNCA and SoftTriple where hit@1 is more or less similar while hit@20 is approximately 0.15 higher. By disabling the proxy update of the original PAL, we observe a drastic performance gain where hit@1 improved by approximately 0.06 while hit@20 gained approximately 0.34. We carefully infer that such performance gains are due to 1) having the fixed proxies providing a consistent representation of the two classes throughout the multi-step training procedure, and 2) maximum distance between two proxies allowing more space for data points to form clusters.

Lastly, we see that BPL (oppositely initialized fixed proxies and maximum margin) significantly outperforms other baselines, especially on hit@1. We further investigate the feasibility of BPL by comparing the performance attained when the margin is randomly configured. As Figure 6 illustrates, BPL with random margin (middle) is slightly worse than the BPL with max margin (far right) while performing much better than PAL without the proxy update (far left).

3) The most effective binary likelihood measurement: As discussed in Section 3.D, distance-based approaches are ineffective to estimate the binary likelihood score of the merged traces $y(x_{ij})$. This is due to the fact that proxy-based losses are not specifically designed to measure the relative distance between data points and their corresponding proxies.

To explore the feasibility of different approaches, as shown

Encoder		TCN	TCN	TCN	TCN	GRU	TCN
Loss Function		ProxyNCA	SoftTriple	PAL	PAL	BPL	BPL
Proxy Initialization		Random	Random	Random	Random	Opposite	Opposite
Proxy Update		Yes	Yes	Yes	No	No	No
Margin		Random	Random	Random	Random	Max	Max
Step 1 & 3	hit@1	0.079	0.029	0.051	0.124	0.035	0.416
	hit@5	0.204	0.115	0.153	0.287	0.078	0.633
	hit@10	0.251	0.182	0.218	0.427	0.190	0.727
	hit@20	0.271	0.248	0.268	0.668	0.391	0.821
Step 2 & 3	hit@1	0.006	0.008	0.012	0.024	0.026	0.129
	hit@5	0.028	0.032	0.025	0.083	0.091	0.251
	hit@10	0.039	0.036	0.033	0.126	0.121	0.352
	hit@20	0.067	0.059	0.035	0.162	0.167	0.583
Step 1, 2 & 3	hit@1	0.205	0.179	0.128	0.183	0.080	0.502
	hit@5	0.262	0.243	0.301	0.446	0.201	0.694
	hit@10	0.295	0.260	0.372	0.600	0.294	0.770
	hit@20	0.274	0.272	0.416	0.755	0.453	0.843

TABLE I: Performance of TLink using different configurations.

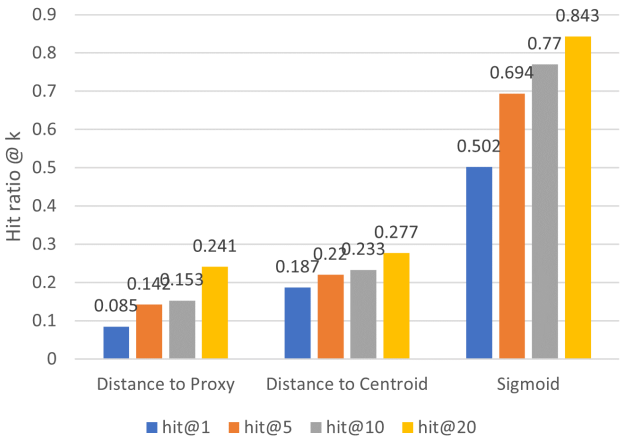


Fig. 7: Performance of different binary likelihood estimation methods.

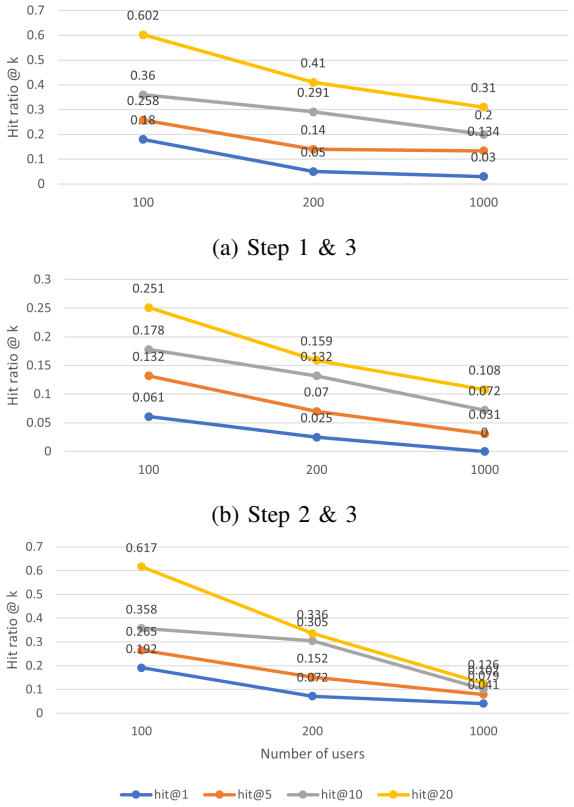


Fig. 8: Performance of DPLink-Lite under different settings.

in Figure 7, we tested three methods including 1) distance between data point and proxy (distance to proxy), 2) distance between data point and centroid (distance to centroid), and 3) score derived by a sigmoid function (sigmoid). For the simplicity of our discussion, we exploited a model using TCN and BPL where proxies are oppositely initialized without update, and margin is set to be maximum.

The result indicate that the distance-based approaches perform very poorly compared to the sigmoid function. For example, the distance to proxy method yielded hit@1 of 0.085 while the sigmoid demonstrated 0.502. To further explore the feasibility of the distance-based approach, we employed a centroid, representing the mean of the positive embedded vectors, for the positive class (traces of the same user) and computed the distance. As the result shows, the distance to centroid approach worked better than that of the proxy, hit@1 of 0.187, however, it is still far below the sigmoid method.

4) Comparing against DPLink-Lite: To make a fair comparison between TLink against DPLink, we made the aforementioned adjustments, described in Section 4.A, to the orig-

inal DPLink to come up with DPLink-Lite. Both TLink and DPLink-Lite followed the same multi-step training procedure and configurations. Note that the original DPLink model was evaluated on datasets with a relatively small number of users (200 to 300). In fact, it is worth investigating how the performance varies as we increase the number of users $|X_t|$. The results shown in Figure 8 suggest that DPLink-Lite is far below the performance of TLink. For example, hit@1 of DPLink-Lite with 1,000 users under the complete training steps (1, 2 & 3) is found to be 0.041 while TLink under the same setting yielded 0.502. While DPLink-Lite performed relatively better compared to itself when the number of users $|X_t|$ was smaller (100 and 200), this is not favorable since we are looking for a solution that can address a larger user pool. Such a low performance of DPLink-Lite is due to missing the pre-training phase the original DPLink required. As mentioned by the authors of DPLink, the pre-training step using the historical data has a significant impact on the model. In addition, training DPLink without location semantics would cause a negative impact on the model as well.

5) Different encoders: RNN vs. TCN: Lastly, to verify the advantage of the TCN-based encoder, we replaced the encoder shown in Figure 5 with GRU of hidden size 128. Apart from the already expected slower convergence and longer training time, the GRU-based approach performs worse than the TCN-based model as shown in Table 1 (second to the last column and the last column). For example, the GRU-based model trained using the complete multi-step procedure (1, 2 & 3) shows hit@1 of 0.080 while the TCN-based model yields 0.502.

V. Conclusion

In this paper, we proposed TLink, a deep learning framework that links mobility traces of the same user across different mobility datasets. TLink encodes the mobility traces using the temporal convolutional network, namely 1D-ResNet, and measures the binary likelihood score using sigmoid function. To effectively train the model, we introduced the binary proxy anchor loss and multi-step training procedure. Our evaluation results illustrated the advantages of TLink over existing methods. In the future, we plan to explore the impact of different levels of data sparsity and the effectiveness of other TCNs.

References

[1] Q. Gao, F. Zhou, K. Zhang, G. Trajcevski, X. Luo, and F. Zhang, "Identifying human mobility via trajectory embeddings." in *IJCAI*, vol. 17, 2017, pp. 1689–1695.

[2] X. Luo, S. Li, and Y. Peng, "Cnntop: a cnn-based trajectory owner prediction method," *ArXiv*, vol. abs/2001.01185, 2020.

[3] C. Miao, J. Wang, H. Yu, W. Zhang, and Y. Qi, "Trajectory-user linking with attentive recurrent network," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 878–886.

[4] Y. Yu, H. Tang, F. Wang, L. Wu, T. Qian, T. Sun, and Y. Xu, "Tulsn: Siamese network for trajectory-user linking," *2020 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8, 2020.

[5] F. Zhou, Q. Gao, G. Trajcevski, K. Zhang, T. Zhong, and F. Zhang, "Trajectory-user linking via variational autoencoder." in *IJCAI*, 2018, pp. 3212–3218.

[6] J. Feng, M. Zhang, H. Wang, Z. Yang, C. Zhang, Y. Li, and D. Jin, "Dplink: User identity linkage via deep neural network from heterogeneous mobility data," in *The World Wide Web Conference*, 2019, pp. 459–469.

[7] H. Ismail Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data mining and knowledge discovery*, vol. 33, no. 4, pp. 917–963, 2019.

[8] S. Kim, D. Kim, M. Cho, and S. Kwak, "Proxy anchor loss for deep metric learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3238–3247.

[9] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 1735–1742.

[10] E. Hoffer and N. Ailon, "Deep metric learning using triplet network," in *International workshop on similarity-based pattern recognition*. Springer, 2015, pp. 84–92.

[11] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," in *Advances in neural information processing systems*, 2016, pp. 1857–1865.

[12] Y. Movshovitz-Attias, A. Toshev, T. K. Leung, S. Ioffe, and S. Singh, "No fuss distance metric learning using proxies," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 360–368.

[13] Q. Qian, L. Shang, B. Sun, J. Hu, H. Li, and R. Jin, "Softtriple loss: Deep metric learning without triplet sampling," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6450–6458.

[14] B. Ghojogh, A. Ghodsi, F. Karray, and M. Crowley, "Spectral, probabilistic, and deep metric learning: Tutorial and survey," *ArXiv*, vol. abs/2201.09267, 2022.

[15] J. Zhang, X. Kong, and P. S. Yu, "Transferring heterogeneous links across location-based social networks," in *Proceedings of the 7th ACM international conference on Web search and data mining*, 2014, pp. 303–312.