T-CONV: A Convolutional Neural Network For Multi-scale Taxi Trajectory Prediction

Jianming Lv
School of Computer Science and Engineering
South China University of Technology
Email: jmlv@scut.edu.cn

Qing Li
City University of Hongkong
Email:qing.li@cityu.edu.hk

Qinghui Sun
School of Computer Science and Engineering
South China University of Technology
Email:csqinghuisun@mail.scut.edu.cn

Xintong Wang
School of Computer Science and Engineering
South China University of Technology
Email:csalexwang@mail.scut.edu.cn

Abstract—Precise destination prediction of taxi trajectories can benefit many intelligent location based services such as accurate ad for passengers. Traditional prediction approaches, which treat trajectories as one-dimensional sequences and process them in single scale, fail to capture the diverse two-dimensional patterns of trajectories in different spatial scales. In this paper, we propose T-CONV which models trajectories as two-dimensional images, and adopts multi-layer convolutional neural networks to combine multi-scale trajectory patterns to achieve precise prediction. Furthermore, we conduct gradient analysis to visualize the multi-scale spatial patterns captured by T-CONV and extract the areas with distinct influence on the ultimate prediction. Finally, we integrate multiple local enhancement convolutional fields to explore these important areas deeply for better prediction. Comprehensive experiments based on real trajectory data show that T-CONV can achieve higher accuracy than the state-of-the-art methods.

Keywords-trajectory; multi-scale; convolutional neural network;

I. INTRODUCTION

Taxi has become one of the major transportation tools in big cities nowadays. For efficient schedule and security monitoring of taxis running in a city, mobile GPS devices are widely installed in most of taxis to record and report their trajectories. Analysis of the destinations of taxi trajectories can benefit a lot of location based services for passengers, such as recommending sightseeing places, accurate ad based on destinations, etc.

Fig. 1 shows a typical use case of the destination prediction, in which a mobile TV installed in the running taxi reports its location to an ad company at fixed time intervals. Based on the trajectory of the taxi and the knowledge learned from the historical trajectories stored in the database, the company predicts that the destination of this trip may be close to a shopping mall with high probability, and thus it pushes some discount information of the products currently on sale at the mall to the mobile TV. How to precisely
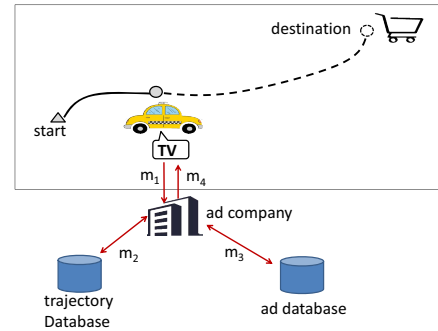

Figure 1. A use case of the destination prediction problem: pushing ad to a running taxi based on its predicted destination. m_1: the running taxi reports its trajectory to the ad company. m_2: the company predicts that the destination of the taxi is close to a shopping mall based on the historical trajectory database. m_3: the company queries the ad database for the ad content related to the destination. m_4: the company pushes the ad to the TV terminal installed in the taxi.

predict the destination of a running taxi based on historical trajectories is the key problem of this kind of location based recommendation tasks.

The most recent event related to this prediction problem is the ECML-PKDD competition [1], which was set up to predict the destinations of taxi trajectories in Porto. The winner of the championship [2] adopted MLP (multi-layer perception) and RNN (recurrent neural network) algorithms. In addition, some recently proposed models [3–5] predict destination by using Markov models to match query trajectories with historical records, or cluster the moving objects according to their trajectories and makes prediction according to the mobility patterns of similar objects[6].

Most of existing researches model trajectories as sequences of spatio-temporal points or road segments, and process them in single spatial scale. This kind of one-

dimensional data structures is hard to explicitly reveal the two-dimensional spatial features of trajectories, e.g. crossings, corners, and windings, which are highly related to the structure of road networks and very useful to understanding of the state and destination of taxi trajectories. Furthermore, we observe that trajectories usually have distinct multi-scale properties, which prefer a multi-scale algorithm to improve the prediction performance. In particular, trajectories in different spatial scale can exhibit different patterns. In a micro scale, a trajectory can be observed with precise positions on the road network. However, in this scale the side-effect of noise has great impact and it is hard to discover the global trend of a large spatial scope. On the contrary, in a macro scale, the global trend of a trajectory can be easily revealed, whist its resolution is relatively low and many details are lost to support accurate prediction. In fact, combinations of the patterns in different scales may help capture the motion features more precisely and completely for better prediction.

Based on the above observation, we propose a novel prediction algorithm, T-CONV, which models trajectories from a different view by considering them as two-dimensional images, and applies multi-layer convolutional neural networks (CNN) to extract multi-scale two-dimensional trajectory features for accurate prediction. Comprehensive experiments based on real data [1] show that our model can achieve better precision than state-of-the-art algorithms.

The main contributions of this paper are summarized as follows:

- For accurate prediction, T-CONV models trajectories as two-dimensional images and adopts multi-layer convolutional neural networks to combine multi-scale two-dimensional trajectory features. Compared with traditional one-dimensional trajectory models, T-CONV is more explicit and efficient to capture two-dimensional local patterns in different spatial scales.

- To visualize the multi-scale patterns captured by T-CONV, the gradient distribution of features is analyzed, and the results show that T-CONV can ① learn small-scale trajectory patterns effectively in its lower convolutional layers, and ② combine them into large-scale patterns in higher layers. The visualization also reveals that the portions close to the start and the end locations of an input trajectory have distinct impact on the prediction of its ultimate destination.

- Beyond convolving the whole input trajectory, we further integrate multiple local enhancement convolutional fields in T-CONV to further explore the specific important areas in a trajectory image, which not only can reduce the sparsity problem in trajectory datasets, but also may greatly improve the accuracy of prediction.

- We conduct comprehensive experiments based on real trajectory data, the results of which show that T-CONV can achieve better accuracy than state-of-the-art methods.

The following sections are organized as follows. Section II describes some related work. Section III offers problem analysis. Section IV presents T-CONV. Section V shows experimental results to validate T-CONV. We conclude this paper in Section VI.

II. RELATED WORK

In this section, we introduce and review some research works related to trajectory prediction.

Most of the existing research works model a trajectory as a one-dimensional spatio-temporal sequence and predict the destination by matching the query trajectory with historical trajectories. Specifically, [7–9] represent a map as a two-dimensional grid in a unified and fixed spatial scale, and all spatial points in one cell are considered as one location identified by the id of the cell. In this way, each trajectory can be modeled as a sequence of cell ids. These algorithms apply the Bayesian inference method to measure the probability of a given destination conditioned on the observed partial trajectory according to the statistics of the historical records. However, in real deployment, the sparsity problem of trajectories often makes it hard to find the historical trajectories exactly matching the query one. Simply adopting Bayesian inference as [7–9] may return zero probability for all candidate destinations most of the time. To solve this problem, [3, 4] decompose trajectories into sub-trajectories connecting two adjacent locations, and adopt first-orders Markov model to infer the probability of all candidate destinations. [5] extends the model to consider the difference between destinations and passing-by locations by adopting an absorbing Markov chain model. [10] utilizes the road network information and divides trajectories into road segments; the research aims to predict the future path of a moving object by matching the query trajectory with historical trajectories on a road network. [11] expresses a personal trajectory as a sequence of cells in a grid, and organizes movement patterns in a pattern tree to support prediction.

Furthermore, clustering algorithms are commonly used techniques to overcome the sparsity problem and extract meaningful basic units in trajectories. [6] clusters the moving objects according to their trajectories and makes prediction using the mobility patterns of similar objects. [12] clusters historical trajectories and uses the Gaussian mixture model to describe the distribution of spatial points in a trajectory cluster. Each query trajectory is assigned to multiple clusters and the mean of the destinations of these clusters is the predicted destination. [13] uses clustering algorithms to extract some important spatial points, named the support points, which are close to crossroads on trajectories. An HMM based algorithm is adopted to establish the relationship between support points and destinations. [14] proposes a variant of the DBSCAN clustering algorithm to obtain

stay points, and applies the variable order Markov Model to predict next locations of personal trajectories.

More recently, neural network based algorithms are receiving more attentions, as they can more accurately predict destinations of taxi trajectories; one such algorithm [2] actually won the championship of the ECML-PKDD competition [1]. As reported, [2] considers the first k points from the start and the last k points close to the end of a query trajectory, and feeds these spatial points into neural networks to perform prediction. A lot of neural networks are tested in the research, including normal multi-layer perception, LSTM[15] based RNN(recurrent neural network), bi-directional RNN, and memory networks. Experiments based on a large validation set show that the bi-directional RNN outperforms the other methods.

Most of above studies extract the features of trajectories in only one specific spatial scale. Specifically, in the grid based algorithms [3, 7–9], the scale is determined by the density of the grid, which is usually predefined and unified in the systems. The clustering based algorithms [6, 12–14] cluster the spatial points and trajectories through their spatial distribution, and perform destination prediction based on the clusters which form a specific kind of spatial scale and keep unchanged during the prediction. On the other hand, in the neural network based models such as [2], trajectories are processed in the finest granularity, and prediction is conducted based on the spatial points, each of which is represented as a tuple of latitude and longitude.

III. PROBLEM ANALYSIS

A. Problem Definition

A taxi has two different types of states: the idle state (no passengers) and the occupied state (with passengers), which can be distinguished by the signals collected from the seat occupancy sensors in the taxi. As shown in the m_1 and m_2 steps of Fig. 1, the problem considered in this paper is to predict the destination of a running taxi in the occupied state.

Formally, given a taxi T_i, its no. j trajectory $\zeta_{i,j}$ is usually recorded as a sequence of GPS locations collected from the GPS devices at fixed time intervals:

$$\zeta_{i,j} =< \delta_{i,j,1}, \delta_{i,j,2}, ..., \delta_{i,j,N_{i,j}} > . \tag{1}$$

Here

$$\delta_{i,j,k} = (\Phi_{i,j,k}, \Upsilon_{i,j,k})(1 \leq k \leq N_{i,j}) \tag{2}$$

is a GPS location. $\Phi_{i,j,k}$ and $\Upsilon_{i,j,k}$ are the longitude and latitude of $\delta_{i,j,k}$. $N_{i,j}$ is the length of the trajectory $\zeta_{i,j}$. Especially, the first location $\delta_{i,j,1}$ means the start of the current trip, which indicates where the passenger(s) came from. The last location $\delta_{i,j,N_{i,j}}$ is the newest location of the taxi. The ultimate destination of any trajectory $\zeta_{i,j}$ is denoted by $\Theta(\zeta_{i,j})$.

The prediction problem can be defined as follows: given any trajectory $\zeta_{i,j}$ of a running taxi T_i, we need to predict
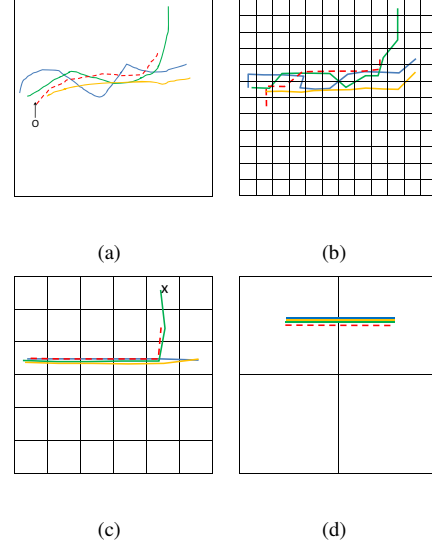

Figure 2. Trajectories in different spatial scales. (a) Original trajectory. (b-d) Trajectories showed in the grids of different spatial scale.

the ultimate destination $\Theta(\zeta_{i,j})$ of the taxi in the current trip based on the knowledge learned from the historical trajectory set.

B. Multi-scale Properties of Trajectories

While trajectories are displayed on the map, a two-dimensional image such as Fig. 2(a) can be obtained. Similar to other two-dimensional spatial data such as weather and traffic data, trajectories also have significant multi-scale properties: in different spatial scales, trajectories can exhibit different two-dimensional patterns. Fig. 2 shows a typical example of taxi trajectories in different scales. The dash line represents the trajectory $\zeta_{i,j}$ of a running taxi starting from the location marked with 'O'. Other trajectories in the figures are historical trajectories in the training set. Fig. 2(a) shows the original trajectories in the highest resolution. When processing trajectories in different scales, the grid is a commonly used tool to divide the map and combine the spatial points belonging to the same cell into one indivisible unit. Fig. 2(b) shows the micro spatial scale with a dense grid, in which trajectories are recorded with fine granularity. In this scale, the view of trajectories displays most details of their motion patterns, and the overlapping degree of the trajectories is low, which causes the sparsity problem when performing trajectory matching. As we divide the map into a sparser grid, we can illustrate the trajectories in a larger spatial scale, as shown by Fig. 2(c). In this scale, the overlapping of the trajectories increases and the global trend of moving objects are easier to be captured. The trajectories are grouped into the two clusters, according to which the destination of the taxi looks like to be 'x' with a high probability. Fig. 2(d) shows the macro spatial scale, in which the global trend of trajectories is clearer and all trajectories

overlap, but no detail about the local difference is preserved, which may nevertheless be important for accurate prediction.

In a nutshell, trajectories in small scales show micro local spatial patterns like corners, curves and crossings, but cause sparsity problem. On the other hand, in macro scales they show the global trend clearly but it is difficult to sense important local changes. A proper combination of the trajectory patterns in multiple scales may be better than the existing single scale based methods to capture both the micro and macro spatial patterns for more accurate prediction.

IV. TRAJECTORY MODELING AND PREDICTION

A. Image Based Modeling of Trajectory

In order to extract the multi-scale two-dimensional spatial patterns explicitly, we model trajectories as two-dimensional images instead of traditional one-dimensional sequences [2–6].

Firstly, we divide the map into an $M * M$ grid evenly as in Fig.2(b), where M is a predefined constant representing the resolution of the map. We use $C_{m,n}(1 \leq m, n \leq M)$ to denote the cell in the row m and column n of the grid. Each GPS point $\delta_{i,j,k} = < \Phi_{i,j,k}, \Upsilon_{i,j,k} >$ can be mapped to a cell $C_{x,y}$ according to its latitude and longitude. The mapping relationship is denoted as:

$$\delta_{i,j,k} \triangleright C_{x,y}. \tag{3}$$

In this way, given a trajectory $\zeta_{i,j}$ we can achieve a two-dimensional $M * M$ image $I_{i,j}$, where the value of the pixel $I_{i,j}(m,n)(1 \leq m, n \leq M)$ is defined as:

$$I_{i,j}(m,n) \tag{4}$$
$$= \begin{cases} f(\delta_{i,j,k}) & \text{if } \exists \delta_{i,j,k}(\delta_{i,j,k} \in \zeta_{i,j} \wedge \delta_{i,j,k} \triangleright C_{m,n}) \\ 0 & \text{otherwise} \end{cases}$$

The image $I_{i,j}$ is called the *trajectory image* of $\zeta_{i,j}$ in this paper. Here, the function $f(\delta_{i,j,k})$ returns a non-negative value for the GPS point $\delta_{i,j,k}$ to indicate the location information.

B. Prediction Based on CNN

Inspired by the successful application of the convolutional neural networks (CNN) based algorithms [16, 17], which enable extracting multi-scale features from pixel-level raw data of images, we propose a CNN based model, T-CONV, to capture the multi-scale spatial patterns from trajectory images and combine these patterns for destination prediction.

The model is shown in Fig. 3. Given a query trajectory $\zeta_{i,j}$, it is transformed into an $M * M$ trajectory image $I_{i,j}$ and input into the model. The value of the pixel $I_{i,j}(m,n)(1 \leq m, n \leq M)$ is defined as:

$$I_{i,j}(m,n) \tag{5}$$
$$= \begin{cases} 0 & \text{if } \nexists \delta_{i,j,k}(\delta_{i,j,k} \in \zeta_{i,j} \wedge \delta_{i,j,k} \triangleright C_{m,n}) \\ 1 & \text{if } \delta_{i,j,N_{i,j}} \triangleright C_{m,n} \\ 0.5 & \text{otherwise} \end{cases}$$

In order to distinguish the end point of the trajectory from other points, $I_{i,j}(m,n)$ is set to 1 when the last point $\delta_{i,j,N_{i,j}}$ of the trajectory is in $C_{m,n}$, and it is set to 0.5 for other points in the trajectory.

The input images are fed into the CNN model, which is composed of groups of convolution layers and max-pooling layers. Specifically, the convolution layer L_1 convolves $I_{i,j}$ to generate the multi-channel feature map $F^{(1)}_{C \times M \times M}$ (with C channels), each element of which is:

$$F^{(1)}_{c,m,n} \tag{6}$$
$$= \sigma\left(\sum_{h=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} \sum_{w=-\lfloor k/2 \rfloor}^{\lfloor k/2 \rfloor} \theta_{c,h,w} \cdot I_{i,j}(m+h, n+w) + b_c \right).$$

Here $\theta_{c,h,w}$ is the convolution parameter, b_c is the bias parameter, k indicates the size of the sliding convolution window, and $\sigma(.)$ is the activation function, typically a rectified linear activation RELU: $\sigma(x) = max(x, 0)$. The main contribution of the convolution operations is to extract the local spatial dependency among neighboring spatial points.

The following max-pooling layer L_2 in Fig. 3 down-samples the input feature maps to generate high-level features in a larger scale. It filters out the pixels with less activation values to generate distilled feature maps $F^{(2)}_{C \times M/t \times M/t}$, each element of which is:

$$F^{(2)}_{c,m,n} = \max_{0 \leq h,w \leq t-1} F^{(1)}_{c,m*t+h, n*t+w} \tag{7}$$

Here t indicates the pooling size.

The combination of the convolution and max-pooling operations can be applied to the feature maps multiple times to form a hierarchical CNN, such as the layers $L_1 \sim L_4$ in Fig. 3. Higher-level features extract the patterns in larger scales, which are calculated from the combination of lower-level features in smaller scales. In this way, multi-scale patterns are embedded into the output features of the highest level of the CNN.

The output features of L_4 are then flattened into one-dimensional vector S and fed to the fully connected neural layer L_5. Moreover, the meta data about date and client information are transformed as embedded vectors [2] and also flattened into one-dimensional vector S', which is input to the layer L_5. Each output element of L_5 is calculated as follows:

$$e_i = tanh\left(\sum_j W_{i,j} \times S_j + W'_{i,j} \times S'_j + b_i \right) \tag{8}$$

Here $W_{i,j}$, $W'_{i,j}$, and b_i are the parameters of L_5, which need to be learnt.

In the final steps, similar with [2], we calculate the predicted destination P as the weighted sum of the frequently visited locations, which are the cluster centers achieved by the mean-shift clustering algorithm deployed on the
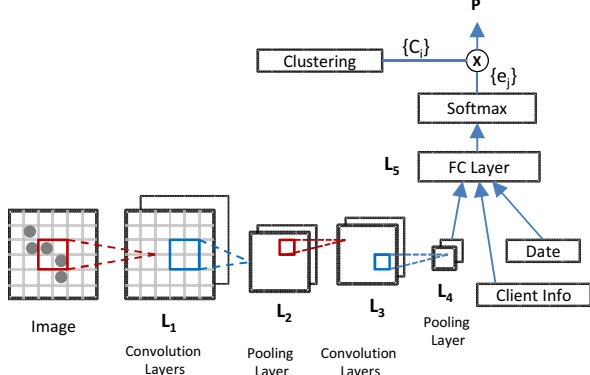
Figure 3. The architecture of the T-CONV model based on CNN.
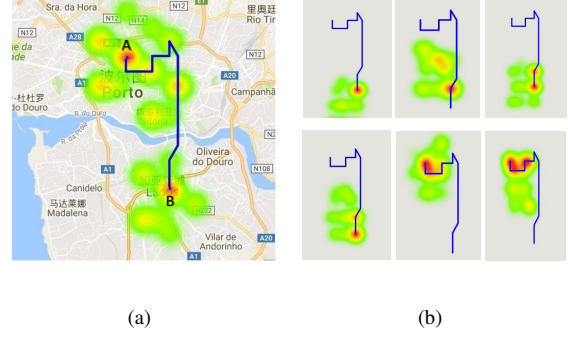

(a)   (b)

Figure 4. (a) The patterns captured in the layer L_4 of T-CONV. The orientation point of the trajectory is the 'A' point and the end point is 'B'. (b)The pattern captured in the layer L_2 of T-CONV.

destinations of all training trajectories. In particular, we have:

$$P = \sum_i c_i \times \frac{exp(e_i)}{\sum_j exp(e_j)}. \qquad (9)$$

Here $\{c_i\}$ is the set of cluster centers, and the weight on each cluster center is determined by the softmax function of the output elements $\{e_i\}$ of L_5.

C. Visualization and Distillation of Patterns

In order to deeply understand the semantics of the features captured by the convolutional neural network, we can measure the relationship between the output features and the input trajectory image by calculating the following gradient:

$$g(k,c,m,n,x,y) = \frac{\partial F_{c,m,n}^{(k)}}{\partial I(x,y)} \qquad (10)$$

Here $F_{c,m,n}^{(k)}$ indicates an output feature of the layer $L_k (1 \le k \le 4)$, and $I(x,y)$ is a pixel of the trajectory image. The gradient indicates the contribution of the pixel value $I(x,y)$ on the feature $F_{c,m,n}^{(k)}$. By visualizing the gradient distribution of all pixels in the image, we can observe what patterns captured by the feature $F_{c,m,n}^{(k)}$ from the raw input image. This visualization method is similar with the Deconvolution Network presented in [18].

We validate this idea by training the model with the real trajectory set from the ECML-PKDD competition [1], and visualize the patterns captured by the features in the different layers of T-CONV. Fig. 4(a) shows the patterns captured by the convolutional layer L_4 while processing a typical trajectory. This pattern is achieved by firstly selecting the largest output feature of L_4 which has the greatest impact on the prediction result, then calculating the gradients over each pixel by Eq. (10), and showing the gradient distribution on the map. In a similar way, as shown in Fig. 4(b), we select the top 6 output features of the layer L_2 and visualize the patterns captured by them.

Fig. 4(b) shows that layer L_2 can capture the small-scale patterns, which focus on the detailed changing in small local areas. Meanwhile, Fig. 4(a) shows that the higher layer L_4 can combine the lower-level small-scale patterns into effective patterns in a larger scale. Furthermore, the red and thick areas in Fig. 4 correspond to bigger gradients and indicate important portions in the patterns. From Fig. 4(a), it is interesting to find that the trained T-CONV model can "recognize" that the local areas close to the start point and end point of the trajectory contribute much more to the destination prediction. Intuitively, the end part of a trajectory is important because it is closest to the destination, and it reveals the latest trend of the trajectory. Meanwhile, the start portion of a trajectory is also important, as it indicates where the customers come from and represents some inner motivation of the trip.

To further verify the uneven contribution of different portions in a trajectory, we randomly select 1,000 trajectories from the dataset [1] and calculate the gradient distribution of different portions of a trajectory as shown in Fig. 5. The distribution is achieved by calculating the gradients of the largest feature in L_4 over the pixels in each trajectory, and make statistics of the gradients in different portions of a trajectory. It shows clearly that the first 10% and last 10% portions of a trajectory have great impact on the destination prediction. This observation motivates us to design better prediction models by further exploring these two important local areas.

D. Local Enhancement CNN for Prediction

Based on the observation that the local portions near the start and end of a trajectory play more significant roles to the destination prediction, we can convolve these local areas deeply beyond deploying convolution over the global image. Fig. 6 shows the architecture of this model, which focuses on the two local areas centered at the start and the end of an input trajectory. Each local area is then divided into an $M * M$ grid and recorded as two $M * M$ trajectory images: *latitude image* I_a and *longitude image*
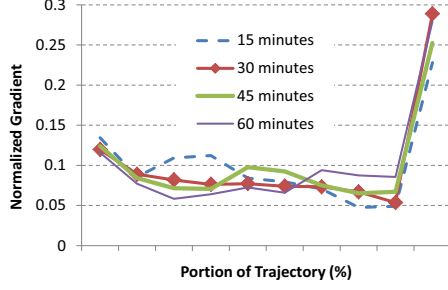
Figure 5. The gradient distribution of different portions of a trajectory. Here L denotes the time duration of the trajectories in minutes.
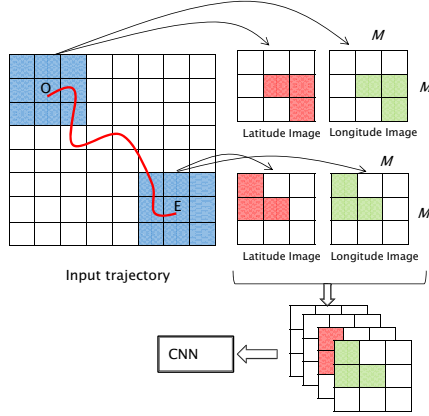

Figure 6. The architecture of the local enhancement CNN. In this model, the two local areas which are close to the start location 'O' and the end location 'E' of the trajectory are convolved deeply.

I_o, which record the precise latitudes and longitudes of the points in the sub-trajectory belonging to the local area. Specifically, given the local area, which contains the sub-trajectory $\zeta'_{i,j} = < \delta'_{i,j,1}, \delta'_{i,j,2}, ..., \delta'_{i,j,n} >$, the pixel value of its latitude image I_a is assigned as:

$$I_a(m,n) = \begin{cases} Norm(\Phi(C_{m,n})) & \text{if } \exists \delta'_{i,j,k}(\delta'_{i,j,k} \rhd C_{m,n}) \\ 0 & \text{otherwise} \end{cases}$$

. The pixel value of its longitude image I_o is assigned as follows:

$$I_o(m,n) = \begin{cases} Norm(\Upsilon(C_{m,n})) & \text{if } \exists \delta'_{i,j,k}(\delta'_{i,j,k} \rhd C_{m,n}) \\ 0 & \text{otherwise} \end{cases}$$

. Here $\Phi(C_{m,n})$ means the latitude of the center of the cell $C_{m,n}$, while $\Upsilon(C_{m,n})$ means the longitude of the same position. $Norm(.)$ denotes the normalization function to transform its input into the range [-1,1]. With the above processing, totally four images can be obtained, which

are stacked together and fed into the convolutional neural network of Fig. 3.

The local enhancement convolution on the important local areas may benefit efficient extraction of important patterns for prediction. Furthermore, by only focusing on small subsets of the whole trajectory image, the overlapping degrees of input latitude images and longitude images among different trajectories are much higher than original global trajectory images. This can help overcome the sparsity problem of trajectories [4], and facilitate the convergence of the learning procedure.

V. EVALUATION

In this section, we conduct experiments to validate the effectiveness of the T-CONV models, and compare them with the other state-of-the-art algorithms.

A. Experiment setup

We test the performance of T-CONV on the real trajectory dataset of the ECML-PKDD competition[1], which is collected from 442 taxis running in the city of Porto for a complete year (from 2013-07-01 to 2014-06-30), and has totally 1.7 millions complete trajectories. In order to validate the prediction accuracy, we use the same testing dataset as [2], which contains 19,770 incomplete trajectories randomly selected from the original training set. The remaining trajectories are used for training.

Two T-CONV models are implemented: the *T-CONV-Basic* which means the simple CNN model presented in section IV.B, and the *T-CONV-LE* which means the local enhancement CNN model mentioned in section IV.D. We mainly compare T-CONV with the neural networks based models[2], which win the champion of the ECML-PKDD competition[1]. These models are listed as follows:

Table I
THE EVALUATION ERROR OF MODELS

Model	Error
MLP	2.81
Bidirectional RNN	3.01
Bidirectional RNN with window	2.60
Memory network	2.87
T-CONV-Basic	**2.70**
T-CONV-LE	**2.53**

- *MLP*: a simple multi-layer perceptron model. The input includes the latitudes and longitudes of the first and last 5 points of an input trajectory.
- *Bidirectional RNN*: a bidirectional recurrent neural network which considers a trajectory as a sequence, and each GPS point in the trajectory forms a transition state of RNN.
- *Bidirectional RNN with window*: a variant of the above Bidirectional RNN model, which uses a sliding window of 5 successive GPS points as a transition state of RNN.

- *Memory network*: another variant of the above RNN model which encodes the trajectories into vectors with fixed length, and compares the similarity between the input trajectories and historical trajectories in this vector space.

The evaluation error to measure these models is the Haversine distance between the real destination and the predicted location. The distance is defined as:

$$d(x,y) \quad = \quad 2 \cdot r \cdot \arctan(\sqrt{\frac{a}{1-a}}) \qquad (11)$$

where

$$a = \sin^2(\frac{\phi_x - \phi_y}{2}) + \cos\phi_x \cos\phi_y \sin^2(\frac{\lambda_x - \lambda_y}{2}) \quad (12)$$

, ϕ is the latitude, λ is the longitude and r is the radius of the earth.

B. Performance

The prediction errors of our proposed models are compared with the baseline models listed in Table 1. The result shows that both T-CONV models are better than the baseline MLP models. As shown in Fig. 3, T-CONV models share similar output layers as the MLP layers, but use the multi-layer convolutional neural networks to capture trajectory patterns. The superior performance of T-CONV demonstrates that the effectiveness of this kind of multi-scale patterns extracted by T-CONV. In addition, the T-CONV models are also better than the Bidirectional RNN model and Memory network, which are also based on one spatial scale.

When looking into the two T-CONV models, T-CONV-LE is much better than T-CONV-Basic, which shows that the convolution on the important local areas can enhance the ability of CNN to capture significant patterns. As shown in Fig. 7, we also test the prediction performance of these two models given the trajectories with different completeness ratios. The completeness ratio indicates the percentage of the current observed trajectory compared to its full trajectory, and it implies how far the end point of the trajectory to destination. Fig. 7 shows that the closer of the taxi to the destination, the lower prediction error to be achieved. Furthermore, it also shows that T-CONV-LE performs much better in most of the cases.

C. Sensibility of parameters

In this section, we test how the change of the hyper parameters in T-CONV-LE may affect its prediction efficiency. We focus on the two parameters related to the spatial scale of trajectory processing, namely, M and W:

- M: As shown in Fig. 6, M is the number of rows (or columns) of the grid. A Higher M corresponds to a denser grid with small granularity of the cells.
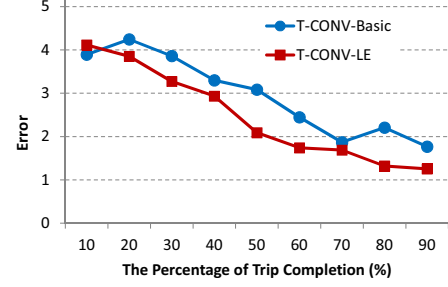

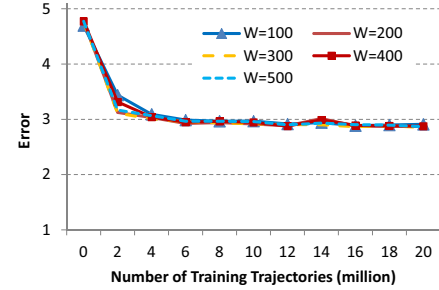Figure 7. The prediction error given different completeness ratio of trajectories

Figure 8. The prediction error of the T-CONV-LE model with different W, which is the width of each cell. M is kept fixed as 30 here.

- W: Measured in meters, W is the width of each cell of the grid. Thus $M * W$ indicates the size of the local areas of T-CONV-LE.

We test the performance of T-CONV-LE with different combinations of various W and M. Fig. 8 and Fig. 9 show how the prediction error decreases during the training stage. Specifically, Fig. 8 shows the performance of the models with different W while M is unchanged, and Fig. 9 shows the performance of the models with different M given fixed W. Both Fig. 8 and Fig. 9 indicate that the performance of T-CONV-LE is not sensitive to the slight change of W and M. Thus, the multi-layer convolutional neural networks of the model are robust to extracting multi-scale patterns for precise prediction.

VI. CONCLUSIONS

In this paper, we have presented a convolutional neural network based model, T-CONV, to predict the destinations of taxi trajectories. Different from traditional research works which process trajectories as one-dimensional sequences in one single scale, T-CONV models trajectories as two-dimensional images and combines multi-scale trajectory patterns through multi-layer convolution operations. Furthermore, inspired by the observation that different portions of a trajectory have significant difference of contribution to the final prediction, a local enhancement T-CONV model named
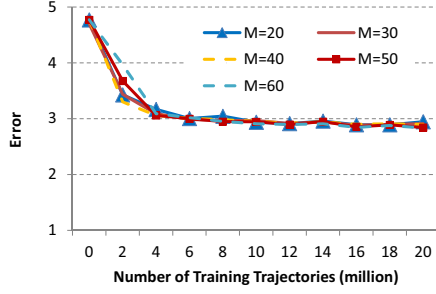
Figure 9. The prediction error given different M, which is the number of the rows in the grid. Here W is kept fixed as 100.

T-CONV-LE is presented to efficiently extract the patterns of the important local areas in a trajectory. Experiments show that T-CONV-LE can achieve better prediction accuracy than state-of-the-art algorithms.

In the future, we will extend the current T-CONV-LE model, which has two local enhancement areas with fixed size, to the more general one, which has multiple local enhancement areas with tunable size. We will also extend the experiments to validate our model in more real trajectory datasets.

REFERENCES

[1] Kaggle. Kaggle competition. https://www.kaggle.com/c/pkdd-15-predict-taxi-service-trajectory-i, 2015.

[2] Alexandre de Brébisson, Étienne Simon, Alex Auvolat, Pascal Vincent14, and Yoshua Bengio14. Artificial neural networks applied to taxi destination prediction. *ECML-PKDD-DCs*, 2015.

[3] Andy Yuan Xue, Rui Zhang, Yu Zheng, Xing Xie, Jin Huang, and Zhenghua Xu. Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In *IEEE ICDE*, 2013.

[4] Andy Yuan Xue, Jianzhong Qi, Xing Xie, Rui Zhang, Jin Huang, and Yuan Li. Solving the data sparsity problem in destination prediction. *The VLDB Journal*, 24(2):219–243, 2015.

[5] Xiang Li, Mengting Li, Yue-Jiao Gong, Xing-Lin Zhang, and Jian Yin. T-desp: Destination prediction based on big trajectory data. *IEEE Transactions on Intelligent Transportation Systems*, 17(8):2344 – 2354, 2016.

[6] Meng Chen, Yang Liu, and Xiaohui Yu. Predicting next locations with object clustering and trajectory clustering. In *PAKDD*, pages 344–356. Springer, 2015.

[7] John Krumm and Eric Horvitz. Predestination: Inferring destinations from partial trajectories. In *Springer UbiComp*, 2006.

[8] Ling-Yin Wei, Yu Zheng, and Wen-Chih Peng. Constructing popular routes from uncertain trajectories. In *ACM SIGKDD*, 2012.

[9] Brian D Ziebart, Andrew L Maas, Anind K Dey, and J Andrew Bagnell. Navigate like a cabbie: Probabilistic reasoning from observed context-aware behavior. In *ACM UbiComp*, 2008.

[10] Sang-Wook Kim, Jung-Im Won, Jong-Dae Kim, Miyoung Shin, Junghoon Lee, and Hanil Kim. Path prediction of moving objects on road networks through analyzing past trajectories. In *International Conference on Knowledge-Based and Intelligent Information and Engineering Systems*, pages 379–389. Springer, 2007.

[11] Ling Chen, Mingqi Lv, and Gencai Chen. A system for destination and future route prediction based on trajectory mining. *Pervasive and Mobile Computing*, 6(6):657–676, 2010.

[12] Philippe C Besse, Brendan Guillouet, Jean-Michel Loubes, and François Royer. Destination prediction by trajectory distribution based model. *arXiv preprint arXiv:1605.03027*, 2016.

[13] Juan Antonio Alvarez-Garcia, Juan Antonio Ortega, L Gonzalez-Abril, and Francisco Velasco. Trip destination prediction based on past gps log using a hidden markov model. *Expert Systems with Applications*, 37(12):8166–8171, 2010.

[14] Jie Yang, Jian Xu, Ming Xu, Ning Zheng, and Yu Chen. Predicting next location using a variable order markov model. In *ACM SIGSPATIAL IWGS*, 2014.

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[16] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[18] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Springer ECCV*, 2014.