PoPPL: Pedestrian Trajectory Prediction by LSTM With Automatic Route Class Clustering

Hao Xue, *Student Member, IEEE*, Du Q. Huynh, *Senior Member, IEEE*, and Mark Reynolds, *Member, IEEE*

***Abstract*— Pedestrian path prediction is a very challenging problem because scenes are often crowded or contain obstacles. Existing state-of-the-art long short-term memory (LSTM)-based prediction methods have been mainly focused on analyzing the influence of other people in the neighborhood of each pedestrian while neglecting the role of potential destinations in determining a walking path. In this article, we propose classifying pedestrian trajectories into a number of route classes (RCs) and using them to describe the pedestrian movement patterns. Based on the RCs obtained from trajectory clustering, our algorithm, which we name the prediction of pedestrian paths by LSTM (PoPPL), predicts the destination regions through a bidirectional LSTM classification network in the first stage and then generates trajectories corresponding to the predicted destination regions through one of the three proposed LSTM-based architectures in the second stage. Our algorithm also outputs probabilities of multiple predicted trajectories that head toward the destination regions. We have evaluated PoPPL against other state-of-the-art methods on two public data sets. The results show that our algorithm outperforms other methods and incorporating potential destination prediction improves the trajectory prediction accuracy.**

***Index Terms*— Crowded scenes, deep learning, long short-term memory (LSTM), trajectory clustering, trajectory prediction.**

I. Introduction

The problem of predicting pedestrians' future trajectories has been one of the main research focus areas in the computer vision and artificial intelligence community in very recent years [1]–[7]. The prediction of pedestrians' trajectories in crowded scenes is essential for social robot human-awareness navigation, intelligent tracking, and driverless vehicles. However, due to the crowdedness and the clutter of many scenes, it is very challenging to predict pedestrians' future trajectories through mining a large volume of video data. For instance, humans can easily make predictions about the future possible location of any person in the scene based on his/her current direction and potential obstacles in the scene. However, this task is extremely hard for computers.

Manuscript received June 22, 2018; revised February 16, 2019 and August 25, 2019; accepted February 15, 2020. Date of publication March 10, 2020; date of current version January 5, 2021. The work of Hao Xue was supported by an International Postgraduate Research Scholarship (IPRS) at The University of Western Australia (UWA). *(Corresponding author: Hao Xue.)*

The authors are with the Department of Computer Science and Software Engineering, The University of Western Australia, Perth, WA 6009, Australia (e-mail: hao.xue@research.uwa.edu.au; du.huynh@uwa.edu.au; mark.reynolds@uwa.edu.au).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2020.2975837

In the literature of pedestrian trajectory prediction, there are basically two categories of approaches: model-based methods and deep learning-based methods. In the first category, the focus is on developing human movement patterns and behavior models [1], [8]–[11]. The problem associated with this category of methods is that the trajectory prediction process heavily depends on manually designed energy functions and specific settings of pedestrian properties. The deep learning methods in the second category use training data to learn and predict trajectories. Inspired by the successes in using the long short-term memory (LSTM) in time-series data prediction, the LSTM architecture has been applied in trajectory prediction [2], [3], [5]. More recently, attention mechanisms used in natural language processing (NLP) have also been incorporated with LSTM for time-series prediction [12] and trajectory prediction [13], [14]. So far in the literature, the focus of the methods in this category has been on analyzing the influence of other pedestrians in the neighborhood of the person being considered (also known as a person of interest or POI for short), whereas the influence from the scene layout has not been widely explored. The influence of the scene layout mentioned here refers to, for instance, walking toward an exit or avoiding an obstacle in the scene. Such influence factors, if incorporated into the trajectory learning process, would improve the prediction results. Moreover, as real complex scenes often have several entrances and exits, algorithms that can predict multiple trajectories corresponding to different possible destinations of the pedestrians would be more useful.

Our aim in this article is to generate plausible predicted trajectories of a pedestrian based on his/her previous observed trajectory. Our method falls into the second category described above. We call our pedestrian trajectory prediction method prediction of pedestrian paths by LSTM (PoPPL). It takes into account different pedestrians' moving patterns in the route class (RC) information as well as produces multiple trajectory predictions. Our method comprises two stages. In the first stage, these clustered RCs are used to train a bidirectional LSTM-based network. In the second stage, one of the following LSTM-based networks, referred to as the sub-LSTMs, is trained to predict trajectories for each RC:

1) a default two-layer LSTM with encoder and decoder, which we refer to as PoPPL-def;
2) a one-layer LSTM, referred to as PoPPL-att, which includes an attention mechanism that incorporates the

correlation between the encoder and each prediction step of the decoder to improve the prediction results;
3) a one-layer LSTM, referred to as PoPPL-rca, which employs adaption to the RCs for trajectory prediction.

Our proposed sub-LSTMs for 2) and 3) are inspired by the recent successes in the attention mechanisms [15], [16] and domain adaptation [17] in various applications [18]–[20].

We evaluate PoPPL on two publicly available data sets: the New York Ground Central [21] (NYGC) and the Edinburgh Informatics Forum data set [22] (Edinburgh). As these two data sets include both indoor environment and outdoor environment, they are well-suited for evaluating the generalization performance of prediction methods.

The key contributions of this article are summarized as follows.

1) We introduce an automatic trajectory clustering component in the pedestrian trajectory prediction problem using source/destination clustering which is computationally efficient. Through trajectory clustering, we obtain different RCs which capture different trajectory patterns in the subsequent trajectory prediction process.
2) The clustered RCs are used to train a bidirectional LSTM network in the first stage of PoPPL. This network predicts the RCs to which each observed trajectory would likely belong. Three different sub-LSTM architectures are proposed to generate trajectories in the second stage.
3) Through a two-stage LSTM-based prediction process, PoPPL can generate multiple prediction results corresponding to different RCs.

The rest of this article is organized as follows. Section II presents a review on trajectory clustering methods and pedestrian trajectory prediction methods in the literature. Section III gives a brief review on the LSTM and bidirectional LSTM architecture. In Section IV, we present PoPPL in detail, including how trajectories are clustered. The data sets used for evaluation, details of our experiments, and results are given in Section V. Finally, we conclude this article and discuss future directions in Section V.

II. Related Work

A. Trajectory Clustering

Trajectory data capture the motion characteristics and movement behavior patterns over time. When many trajectories are available, motion patterns can be learned [23] through trajectory clustering. Trajectory clustering has also been applied to pedestrian counting [24], surveillance video analysis [25]–[27], traffic behavior understanding [28], [29], and anomaly detection [30]–[32].

To cluster trajectories, a suitable similarity measure must be determined. The Euclidean distances or the Hausdorff distances are the two most common distance measures for comparing points between trajectories. The most intuitive distance measure is simply using the sum of distances between the corresponding pairs of points [33]. However, this measure requires the number of sample points, commonly referred to as the temporal length of the trajectories to be the same. For trajectories having different temporal lengths, the Euclidean distance is not defined and cannot be calculated directly.

In the early 1990s, dynamic time warping (DTW) was introduced for computing the distance between two unequal temporal length trajectories [34]. Given that DTW is relatively sensitive to noise, the longest common subsequence (LCSS) has been used to overcome this problem in trajectory clustering [35]. Another popular similarity distance measure for comparing trajectories is the edit distance on real sequence (EDR) [36], which is closely related to LCSS. Recently, variants of DTW have been proposed to speed up the traditional DTW algorithm [37], [38]. Atev *et al.* [39] used both DTW and LCSS and compared them in vehicle trajectory clustering for traffic scenes. Their method involves applying the spectral analysis to the affinity matrix computed from a modified Hausdorff distance measure for vehicle trajectories. Although their method worked well on traffic scenes, it is not suitable for pedestrian trajectory clustering as vehicles are constrained to move on the road network, whereas pedestrians in a crowded scene have much more complex movement patterns. Moreover, in crowded scenes such as a train station, where the number of trajectories is large (e.g., in the thousands), these trajectory clustering algorithms are computationally expensive.

B. Model-Based Trajectory Prediction

The simplest method to predict a pedestrian's trajectory is to use his/her current position and walking velocity. Indeed, the walking velocity is one of the basic factors used in model-based prediction methods. In their pioneering research article, Helbing and Molnar [8] proposed a social force (SF) model, which uses different forces to describe human motion patterns. Apart from having a motion force term to capture the acceleration of pedestrians, their model includes two more forces: repulsive force (e.g., repulsion from obstacles) and attractive force (e.g., attraction from potential destinations). Yamaguchi *et al.* [9] extended this model by incorporating more behavior factors such as damping, collision, and social interactions. Using an agent-based model, Yi *et al.* [21] combined a stationary crowd group factor, a moving pedestrian factor, and a scene layout factor together to deal with trajectories of crowded scenes.

Another group of model-based prediction methods involves using the Gaussian process model [40]. In a trajectory, the coordinates at each time step are considered to be a Gaussian random variable that models the distribution of this person's own previous trajectory. An interacting Gaussian processes (IGP) model has been developed to solve the frozen robot problem in a robot crowded scene navigation scenario [41]. Based on the IGP model, an interaction model [6] is designed to describe the cooperative human behavior in crowded scenes. Su *et al.* [5] addressed the crowd motion prediction problem using a social-aware deep Gaussian process model.

A limitation of these model-based prediction approaches is that they need hand-crafted influence factors, manually designed energy functions, and rules for specific scenarios.

In the last two years, research articles in the literature have shifted to automatically detecting and handling these influence factors.

C. Deep Learning-Based Trajectory Prediction

Recurrent neural networks (RNNs), LSTM networks [42], gated recurrent units (GRU) [43], and simple recurrent unit (SRU) [44] have been designed to handle time sequence data. Therefore, these networks are suitable for exploring temporal sequences and often referred to as deep in time [45]. RNN and its variants have been successfully applied to many time-series problems including speech recognition [46]–[48], language translation [49], action recognition [50]–[52], human interaction recognition [53], and image captioning [54], [55].

The trajectories of pedestrians can be considered as time sequence data. Hence, an LSTM model can be used for predicting trajectories of pedestrians. Alahi *et al.* [2] suggested a social LSTM model based on the sequence generation model from [56], which combines the behavior of other people within a large neighborhood. They used one LSTM model for every person in crowded scenes and introduced a social pooling strategy to connect different LSTM models to describe the interaction of people in the same neighborhood. This social-LSTM [2] has later been improved by incorporating an attention model into the LSTM architecture [13], [14]. However, these methods do not have scene context information incorporated in the trajectory prediction process. Lee *et al.* [4] proposed the deep stochastic inverse optimal control RNN encoder–decoder (DESIRE) framework, where the encoder–decoder network is responsible for generating multiple trajectories for each pedestrian and the convolutional neural network (CNN) is used to rank and refine generated trajectories. RNNs have also been used in GPS trajectory prediction applications such as destination prediction [57], public transportation prediction [58], and location prediction problem [59]. In 3-D trajectory prediction, Sun *et al.* [60] proposed a method to predict trajectories based on 3-D LiDar and RGB-D sensors collected data. Their temporal 3DOF-Pose (T-Pose-LSTM) approach predicts both the 2-D positions and orientations of pedestrians.

Bidirectional LSTMs have recently attracted special attention to sequence labeling problems, such as speech recognition [61], translation [62], and handwriting recognition [63]. Unlike the traditional LSTMs, bidirectional LSTM networks [64] use the information from both the forward and backward directions of the input sequences. As a result, there are more connections in these networks.

This article extends our previous work [65], where we first introduced the use of a bidirectional LSTM to the pedestrian trajectory prediction problem. The extension that we accomplish includes automatically clustering of the trajectory RCs and extensive experiments to compare the effect of different numbers of RCs on the prediction performance. Furthermore, we design three different sub-LSTM architectures that are used in stage 2 to generate trajectories. Compared to the social-LSTM method [2] and other methods [13], [14], our method incorporates global scene context information, in particular, the intended destination area of each pedestrian trajectory. By automatically clustering, complex trajectories are classified into different RCs in stage 1 of our method. In stage 2, sub-LSTMs are trained with trajectories falling into specific RCs obtained from the previous stage. Our experiments show that the diversity of trajectories, due to the complex pedestrian movement patterns, makes it difficult for a single standard vanilla LSTM network to learn to produce good prediction results, whereas with several sub-LSTMs of the same network architecture, where each needs to learn only a specific and simpler trajectory pattern, the prediction results are significantly better.

III. LSTM and Bidirectional LSTM

In a basic LSTM network architecture, given an input sequence represented as $(x_1, \ldots, x_T)$, the output sequence y_t can be obtained by iteratively computing the following equations for $t = 1, \ldots, T$:

$$\mathbf{h}_t = \mathrm{LSTM}(\mathbf{h}_{t-1}, x_t; \mathbf{W}) \tag{1}$$
$$y_t = \mathbf{W}_{hy}\mathbf{h}_t + b_y \tag{2}$$

where $\mathbf{W}$ denotes a set of weight matrices involved in the network, b_y denotes the bias vector for the output y_t, and $\mathbf{h}$ denotes the hidden state. Inside the cell LSTM $(\cdot)$, the hidden state is determined by the input gate vector $\mathbf{i}$, forget gate vector $\mathbf{f}$, output gate vector $\mathbf{o}$, and the cell state vector $\mathbf{c}$ via the following equations:

$$\mathbf{i}_t = \sigma(\mathbf{W}_{xi}x_t + \mathbf{W}_{hi}\mathbf{h}_{t-1} + \mathbf{W}_{ci}\mathbf{c}_{t-1} + \mathbf{b}_i) \tag{3}$$
$$\mathbf{f}_t = \sigma(\mathbf{W}_{xf}x_t + \mathbf{W}_{hf}\mathbf{h}_{t-1} + \mathbf{W}_{cf}\mathbf{c}_{t-1} + \mathbf{b}_f) \tag{4}$$
$$\mathbf{c}_t = \mathbf{f}_t\mathbf{c}_{t-1} + \mathbf{i}_t \tanh(\mathbf{W}_{xc}x_t + \mathbf{W}_{hc}\mathbf{h}_{t-1} + \mathbf{b}_c) \tag{5}$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_{xo}x_t + \mathbf{W}_{ho}\mathbf{h}_{t-1} + \mathbf{W}_{co}\mathbf{c}_t + \mathbf{b}_o) \tag{6}$$
$$\mathbf{h}_t = \mathbf{o}_t \tanh(\mathbf{c}_t) \tag{7}$$

where $\mathbf{W}_{ab}$ is the weight matrix from layers a to b; $\sigma(\cdot)$ denotes the sigmoid activation function; and each $\mathbf{b}$ term with a subscript is the bias vector for the appropriate layer.

Given an input sequence $\mathbf{x} = [x_1, x_2, \ldots, x_t, \ldots, x_T]$, it can be divided into two parts at time step t: previous sequence $\mathbf{x}_p$ and future sequence $\mathbf{x}_f$, where

$$\mathbf{x}_p = [x_1, x_2, \ldots, x_{t-1}] \tag{8}$$
$$\mathbf{x}_f = [x_{t+1}, \ldots, x_T]. \tag{9}$$

Unlike the traditional LSTM cells shown in (1) and (2), the bidirectional architecture explores the information of the input sequences in both the forward $\mathbf{x}_f$ and backward $\mathbf{x}_p$ directions (see Fig. 1). In this case, the LSTM cells become

$$\overrightarrow{\mathbf{h}}_t = \mathrm{LSTM}(x_t, \overrightarrow{\mathbf{h}}_{t-1}; \overrightarrow{\mathbf{W}}) \tag{10}$$
$$\overleftarrow{\mathbf{h}}_t = \mathrm{LSTM}(x_t, \overleftarrow{\mathbf{h}}_{t+1}; \overleftarrow{\mathbf{W}}) \tag{11}$$
$$y_t = \mathbf{W}_{\overrightarrow{h}y}\overrightarrow{\mathbf{h}}_t + \mathbf{W}_{\overleftarrow{h}y}\overleftarrow{\mathbf{h}}_t + b_y \tag{12}$$

where, as before, the $\mathbf{W}$ terms denote different weight matrices. Specifically, $\overrightarrow{\mathbf{h}}_t$, $\overrightarrow{\mathbf{W}}$, $\overleftarrow{\mathbf{h}}_t$, and $\overleftarrow{\mathbf{W}}$ are the hidden states and weight matrices of the forward and backward layers, respectively.

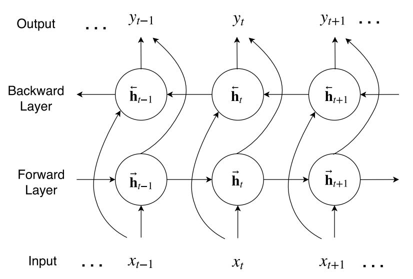

Output
y_{t-1}
y_t
y_{t+1}
Backward Layer
$\overleftarrow{\mathbf{h}}_{t-1}$
$\overleftarrow{\mathbf{h}}_t$
$\overleftarrow{\mathbf{h}}_{t+1}$
Forward Layer
$\overrightarrow{\mathbf{h}}_{t-1}$
$\overrightarrow{\mathbf{h}}_t$
$\overrightarrow{\mathbf{h}}_{t+1}$
Input
x_{t-1}
x_t
x_{t+1}

Fig. 1. Bidirectional LSTM architecture. The forward layer and the backward layer can explore the input sequence from both directions.

IV. Our Method

A. Problem Formulation and System Overview

A brief introduction of the notation that we use is given below. At time instant t, the ith pedestrian in the scene is represented by the image coordinates (x_t^i, y_t^i). We observe the positions of all the pedestrians from time $t = 1$ to $t = T_{\text{obs}}$, and our aim is to predict their positions from $t = T_{\text{obs}} + 1$ to $t = T_{\text{obs}} + T_{\text{pred}}$. Pedestrian trajectory prediction can, therefore, be defined as a sequence generation problem.

Given: Observed trajectories $\mathbf{X}_i^{\text{obs}} = \big[(x_1^i, y_1^i), \ldots, (x_{T_{\text{obs}}}^i, y_{T_{\text{obs}}}^i)\big]\ \forall i$.

Objective: Predict future trajectories $\mathbf{X}_i^{\text{pred}} = \big[(x_{T_{\text{obs}}+1}^i, y_{T_{\text{obs}}+1}^i), \ldots, (x_{T_{\text{obs}}+T_{\text{pred}}}^i, y_{T_{\text{obs}}+T_{\text{pred}}}^i)\big]\ \forall i$.

In real applications, after walking into a shopping square or a train terminal, pedestrians would choose their routes mostly based on their intended destinations. Hence, trajectory prediction will have a much higher accuracy if the intended destination can be learned from the trajectory data. Furthermore, to model the complex movement patterns of pedestrians, rather than a single predicted trajectory for each pedestrian, being able to generate multiple possible trajectories for each pedestrian would be more useful for further analysis such as anomaly detection. Most of the existing trajectory prediction methods would only generate one single predicted trajectory. On the contrary, PoPPL learns different movement patterns and predicts multiple trajectories.

In general, the aim of the first stage of PoPPL is to predict the prospective destinations D_j's and the probability of choosing each D_j. In the second stage, a sequence $\mathbf{X}_{i,j}^{\text{pred}}$ is generated for each destination D_j obtained from the first-stage prediction. Details of RC clustering and the two-stage prediction will be described in Sections IV-B–IV-D.

B. Clustering of RCs

Different movement patterns naturally result in different trajectory clusters. Thus, by separating trajectories into clusters would allow different movement patterns to be dealt with individually. To determine whether two trajectories should be grouped together into the same cluster, the distances between pairs of points from both trajectories must be computed. Trajectory clustering is, therefore, very computationally expensive when the total number of trajectories and the number of points in each trajectory are large.

An RC represents a specific pedestrian movement pattern that comes from one source region and leaves the scene through one destination region. We use $R_{i,j}$ to denote the RC from the source region D_i to the destination region D_j. The problem of clustering the whole trajectories can then be transformed into the task of clustering the source/destination regions. To get the coordinates of points forming the source/destination regions, the starting and ending points of all the trajectories are extracted from the training data set. As pedestrians can walk from a source region A to a destination region B and in the opposite direction from B to A in a crowded scene, we combine all the starting and ending trajectory points into one single-point cloud for clustering. We use the k-means method to obtain N point clusters. Compared to trajectory clustering, this point clustering process is much cheaper computationally and more suitable for trajectories that are generated by pedestrians. Fig. 2 shows a few source/destination point clustering results and some example route classes on two benchmark datasets.

For a scene that has N source/destination regions, theoretically, the total number of RCs is equal to $N(N + 1)/2$. As a sufficient number of trajectories is required to train the LSTM networks in stage 2, our algorithm only selects those RCs, which have their numbers of trajectories equal to at least $\tau_{\min}\%$ of the total number of trajectories in the training set.

C. Stage 1: RC Classification

In the trajectory clustering process described above, the whole trajectories are used to determine the RCs. However, in the trajectory prediction process, only a portion of each trajectory (referred to as the observed trajectory or $\mathbf{X}_i^{\text{obs}}$) is known. As the final destination points of the whole trajectories are unknown, they cannot be directly clustered into RCs.

The aim of this stage is to learn to predict the potential RCs that the observed trajectories would lead to. This is a multiclass classification problem where the number of classes is equal to the number of RCs identified in Section IV-B. The classification task here is slightly different from the traditional classification problems as our overall objective is to "foresee" the possible destination regions of each trajectory. This RC prediction process provides extra information about the POIs' possible future destination regions in the trajectory prediction.

We train a bidirectional LSTM-based classification network, which we refer to as BiLSTM (see Fig. 3), for the multiclass classification problem described above. The network takes the coordinates of the observed trajectories $\mathbf{X}_i^{\text{obs}}$ as input. Hence, at time step t (where $t \le T_{\text{obs}}$) of the bidirectional LSTM cell, the previous sequence $\mathbf{x_p}$ and the future sequence $\mathbf{x_f}$ in (8) and (9) become as follows: the previous information $\mathbf{X}_{i,t}^{\text{previous}} = [(x_1^i, y_1^i), \ldots, (x_t^i, y_t^i)]$ and the future information $\mathbf{X}_{i,t}^{\text{future}} = [(x_{t+1}^i, y_{t+1}^i), \ldots, (x_{T_{\text{obs}}}^i, y_{T_{\text{obs}}}^i)]$. We have $\mathbf{X}_{i,t}^{\text{previous}} \oplus \mathbf{X}_{i,t}^{\text{future}} = \mathbf{X}_i^{\text{obs}}$, where $\oplus$ represents the concatenation operator. Note that $\mathbf{X}_{i,t}^{\text{future}}$ is different from $\mathbf{X}_{i,t}^{\text{pred}}$ at the prediction trajectory stage described in Section IV-A.

After the bidirectional LSTM layers, a merging layer is used to add the two output vectors from the forward and backward LSTM layers. These two output vectors are in

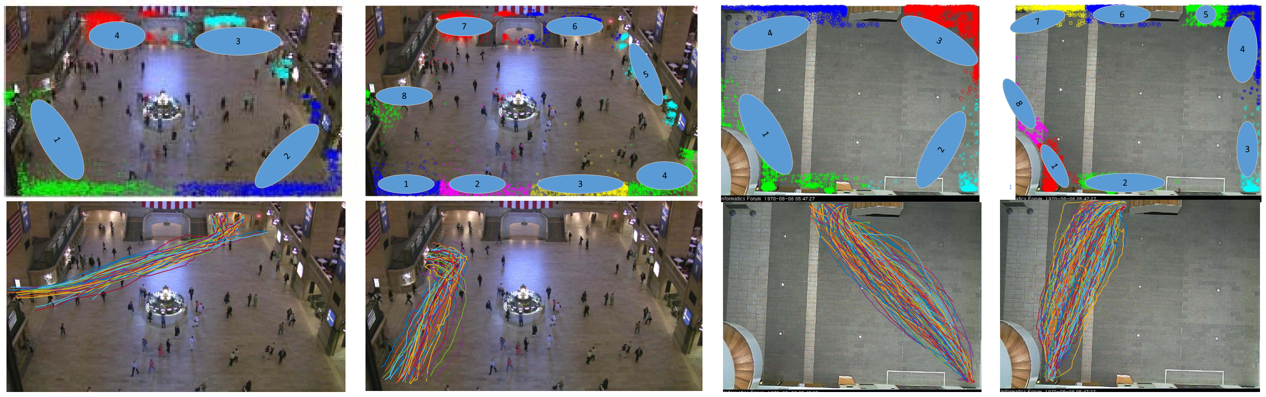


Fig. 2. Results of source/destination point clustering (row 1) and RCs (row 2) on the NYGC and Edinburgh data sets.

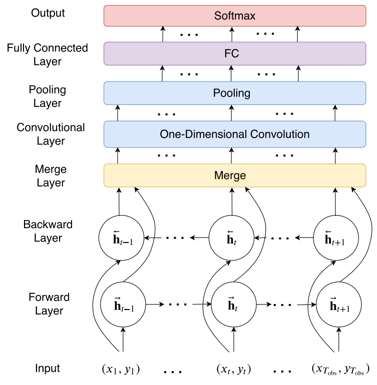

Output
Softmax
Fully Connected Layer
FC
Pooling Layer
Pooling
Convolutional Layer
One-Dimensional Convolution
Merge Layer
Merge
Backward Layer
$\overleftarrow{\mathbf{h}}_{t-1}$
$\overleftarrow{\mathbf{h}}_{t}$
$\overleftarrow{\mathbf{h}}_{t+1}$
Forward Layer
$\overrightarrow{\mathbf{h}}_{t-1}$
$\overrightarrow{\mathbf{h}}_{t}$
$\overrightarrow{\mathbf{h}}_{t+1}$
Input
(x_1, y_1)
(x_t, y_t)
$(x_{T_{\text{obs}}}, y_{T_{\text{obs}}})$

Fig. 3. Stage 1 of PoPPL: BiLSTM, a bidirectional LSTM network for RC classification.

temporal sequence format, and the length is equal to the temporal length of the input observed trajectory (T_{obs}). At each time step of each temporal sequence, the vector size is $\mathbb{R}^r$, where r is the hidden dimension of the LSTM layer. Then, a 1-D convolutional layer as well as a max-pooling layer is applied to a tensor of size $\mathbb{R}^{T_{\text{obs}} \times r}$. Since the purpose of this stage is to classify observed trajectories into RCs, a fully connected layer and a softmax layer are used, so that the outputs of stage 1 are the probabilities of the RCs for which each observed trajectory destines. By training the BiLSTM network shown in Fig. 3, PoPPL can predict these probabilities for new (unseen in the training phase) observed trajectories at the testing phase.

In the training phase of BiLSTM, the trajectory clustering results are used to label the training trajectory set. Specifically, because the network is bidirectional and the source points and the destination points are combined together in the RC clustering process, $R_{i,j}$ and $R_{j,i}$ belong to the same RC. The standard one-hot encoding policy is used to generate labels to represent different RCs. Table I shows an example of six RCs and their one-hot encoding.

D. Stage 2: Generation of Predicted Trajectories

In stage 2 of PoPPL, one of the sub-LSTM network architectures mentioned in Sections I and II can be used

TABLE I
ONE-HOT ENCODING FOR A SCENE THAT HAS SIX RCS

Class No.	Route	Label
1	$R_{1,2}$ and $R_{2,1}$	[0 0 0 0 0 1]
2	$R_{1,4}$ and $R_{4,1}$	[0 0 0 0 1 0]
3	$R_{1,3}$ and $R_{3,1}$	[0 0 0 1 0 0]
4	$R_{2,4}$ and $R_{4,2}$	[0 0 1 0 0 0]
5	$R_{2,3}$ and $R_{3,2}$	[0 1 0 0 0 0]
6	$R_{3,4}$ and $R_{4,3}$	[1 0 0 0 0 0]

for the training and testing phases. For both PoPPL-def and PoPPL-att, n such sub-LSTM networks will be trained if there are n RCs. For the PoPPL-rca network architecture, the training process is slightly different (see later). In the testing phase, given a new observed trajectory, the probabilities of the RCs to which it belongs are used for selecting the trained sub-LSTMs. Specifically, a predefined threshold τ is used, so that only those RCs with probabilities larger than τ will be selected for trajectory prediction.

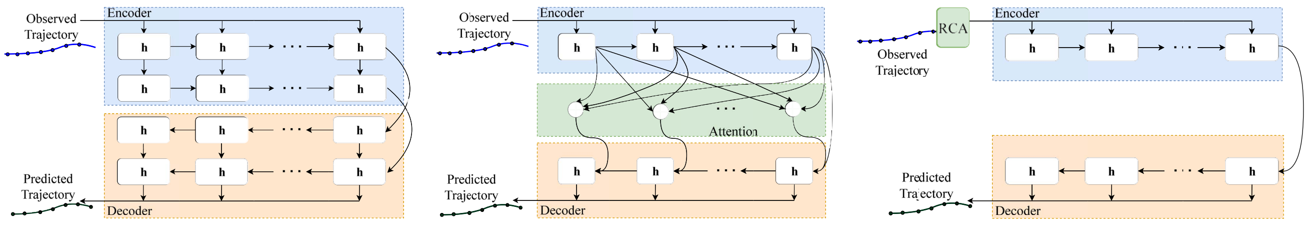
In the remaining part of this section, we will describe the details of the three sub-LSTM network architectures used in stage 2.

1) Default Encoder–Decoder Sub-LSTM (PoPPL-def): This is a general encoder–decoder network having two hidden layers [see Fig. 4(a)], where the LSTM encoder receives an observed trajectory as input and generates a hidden-state sequence. From the hidden-state sequence, the LSTM decoder can generate a predicted trajectory. The input size and the output size of the sub-LSTMs are determined by the length of the observed trajectories and predicted trajectories, respectively. Two LSTM layers are used in the encoder and the decoder to better capture the hidden states of the trajectories passed to the network. For the LSTM encoder, the observed trajectory sequence $\mathbf{X}^i_{\text{obs}}$ of the ith pedestrian is read in as input. The hidden state $\mathbf{h}^i_t$ for this pedestrian at time t is updated via

$$\mathbf{e}^i_t = \phi\left(x^i_t, y^i_t; \mathbf{W}_e\right) \tag{13}$$

$$\mathbf{h}^{i,(1)}_t = \text{LSTM}^{\text{enc}}\left(\mathbf{h}^{i,(1)}_{t-1}, \mathbf{e}^i_t; \mathbf{W}^{(1)}_h\right) \tag{14}$$

$$\mathbf{h}^{i,(2)}_t = \text{LSTM}^{\text{enc}}\left(\mathbf{h}^{i,(2)}_{t-1}, \mathbf{h}^{i,(1)}_t; \mathbf{W}^{(2)}_h\right) \tag{15}$$



(a) PoPPL-def (b) PoPPL-att (c) PoPPL-rca

Fig. 4. Three sub-LSTM architectures implemented in Stage 2 of PoPPL. In (c), RCA stands for the RC adaptation module. Different hidden states **h** shown in the three networks should all have subscripts to denote their time steps. Due to the small figure size, these subscripts are not shown. (a) PoPPL-def. (b) PoPPL-att. (c) PoPPL-rca.

where $\phi(\cdot)$ is the embedding function mapping (x_t^i, y_t^i) to the embedding vector $\mathbf{e}_t^i$; LSTM$^{\text{enc}}$ denotes the LSTM encoder; $\mathbf{W}_e$ is an unknown weight matrix, and $\mathbf{W}_h^{(1)}$ and $\mathbf{W}_h^{(2)}$ are collections of unknown weight matrices; and $\mathbf{h}_t^{i,(k)}$ is the hidden-state vector for layer k, for $k = 1, 2$. The unknown bias terms (not shown in the above equations) and weight matrices are entities that need to be estimated in the training process.

The LSTM decoder, LSTM$^{\text{dec}}$, predicts the sequence of positions $\{(\hat{x}_t^i, \hat{y}_t^i)\}$ of pedestrian i for time $t > T_{\text{obs}}$, via

$$\mathbf{h}_t^{i,(1)} = \text{LSTM}^{\text{dec}}\left(\hat{\mathbf{e}}_t^i, \mathbf{h}_{t-1}^{i,(1)}; \mathbf{W}_d^{(1)}\right) \tag{16}$$

$$\mathbf{h}_t^{i,(2)} = \text{LSTM}^{\text{dec}}\left(\mathbf{h}_t^{i,(1)}, \mathbf{h}_{t-1}^{i,(2)}; \mathbf{W}_d^{(2)}\right) \tag{17}$$

$$\left(\hat{x}_t^i, \hat{y}_t^i\right) = \mathbf{W}_o \mathbf{h}_t^{i,(2)} + \mathbf{b}_o \tag{18}$$

where $\hat{\mathbf{e}}_t^i = \phi(\hat{x}_{t-1}^i, \hat{y}_{t-1}^i$, and $\mathbf{W}_e)$ is the embedding vector obtained from (13) using the predicted coordinates at time $t-1$. Similar to the encoder part, $\mathbf{W}_d^{(k)}$, for $k = 1, 2$, $\mathbf{W}_o$ are unknown weight matrices of the LSTM decoder and the output layer, and $\mathbf{b}_o$ is the unknown bias term of the output layer.

2) Attention Sub-LSTM (PoPPL-att): Given that the attention mechanism is often integrated to improve the performance of LSTM-based models [15], [16], based on the aforementioned basic encoder–decoder, a context vector $\mathbf{c}_t$ is introduced to this sub-LSTM architecture [see Fig. 4(b)] at each time step $t = T_{\text{obs}} + 1, \ldots, T_{\text{obs}} + T_{\text{pred}}$ to capture different attentions of the input sequence of the POI. Thus, (16) evolves into

$$\mathbf{h}_t^i = \text{LSTM}^{\text{dec}}\left(\hat{\mathbf{e}}_t^i, \mathbf{h}_{t-1}^i, \mathbf{c}_t^i; \mathbf{W}_d\right) \tag{19}$$

where the context vector $\mathbf{c}_t^i$ is a weighted sum of all the hidden states in the input observed trajectory sequence of the ith POI, that is

$$\mathbf{c}_t^i = \sum_{s=1}^{T_{\text{obs}}} \alpha_{s,t}^i \mathbf{h}_s^i \tag{20}$$

where $\alpha_{s,t}^i$ is the weight of pedestrian i at time step s of the observed part and t stands for a time step in the predicted part of the trajectory. It can be computed as follows:

$$\alpha_{s,t}^i = \frac{\exp\left(e_{s,t}^i\right)}{\sum_{k=1}^{T_{\text{obs}}} \exp\left(e_{k,t}^i\right)} \tag{21}$$

where $e_{s,t}^i$ is a suitable score function to score how the two parts of the trajectory are correlated. In this article, we adopt the general score function proposed by [16]

$$e_{s,t}^i = \mathbf{h}_s^\top \mathbf{W}_a \mathbf{h}_t \tag{22}$$

where $\mathbf{W}_a$ is a weight matrix that needs to be trained.

Note that our PoPPL-att does not require other pedestrians' information and the temporal attention in PoPPL-att focuses on the POI's own history trajectory only. In this regard, our attention mechanism is different from the attention mechanisms used in [13] and [14].

3) RCA Sub-LSTM (PoPPL-rca): In this sub-LSTM architecture, a module named RCA is introduced [shown in Fig. 4(c)]. Like the concept in domain adaptation, the main idea in PoPPL-rca is to train an LSTM network for predicting trajectories belonging to a specific RC, which we refer to as the base RC. For any of the other RCs, the role of its RCA module is to adapt the trajectories in this RC, so that the network trained on the base RC can be used for prediction. Thus, if there are n RCs, only one LSTM network, which handles the trajectory prediction, and $n-1$ RCA modules need to be trained. The trajectory prediction network for the base RC can be a two-layer LSTM [see Fig. 4(a)], a one-layer LSTM with attention [see Fig. 4(b)], or any other network. In our realization of PoPPL-rca, we use a simple one-layer LSTM with the standard encoder–decoder.

The training phase of this sub-LSTM architecture consists of two parts. In part 1, the RC with the largest number of training trajectories is selected as the base RC and the prediction network is trained using all these trajectories without including the RCA module. In part 2 of the training phase, the weights of the trained prediction network are frozen. An RCA module is now trained using the training trajectories of each of the remaining RCs.

The role of each RCA module is very simple. Suppose that $\widetilde{R}$ is the base RC whose LSTM prediction network has been trained and R is any arbitrary RC. Then, we can train the RCA$^{R\to\widetilde{R}}$ module, so that

$$\widetilde{\mathbf{X}}^{\text{obs}} = \text{RCA}^{R\to\widetilde{R}}(\mathbf{X}^{\text{obs}}). \tag{23}$$

Here, we use all the observed trajectories $\mathbf{X}^{\text{obs}}$ belonging to RC R and map them to some fictitious trajectories $\widetilde{\mathbf{X}}^{\text{obs}}$ in RC $\widetilde{R}$ such that the average displacement error (ADE) (see Section V-C) of the predicted trajectories is minimized.

In our proposed PoPPL-rca, each RCA module comprises three fully connected layers with $2T_{\text{obs}}$, $4T_{\text{obs}}$, and $2T_{\text{obs}}$ neurons, respectively. The advantages of this PoPPL-rca are

Algorithm 1 Trajectory Prediction Process Using PoPPL

```
Input: observed trajectory $\mathbf{X}^{\mathrm{obs}}$ of a POI; trained BiLSTM; $m$ trained sub-LSTMs; threshold $\tau$
Output: $\mathbf{X}^{\mathrm{pred}}$: set of predicted trajectories; $\mathbf{P}^{\mathrm{pred}}$: set of corresponding probabilities
1: $\mathbf{X}^{\mathrm{pred}} \leftarrow \{\}$, $\mathbf{P}^{\mathrm{pred}} \leftarrow \{\}$
2: for $1 \le j \le m$ do
3:     $p_j \leftarrow \mathrm{BiLSTM}(\mathbf{X}^{\mathrm{obs}})$
4:     if $p_j \ge \tau$ then
5:         $\mathbf{X}_j^{\mathrm{pred}} \leftarrow \text{sub-LSTM}_j(\mathbf{X}^{\mathrm{obs}})$
6:         $\mathbf{X}^{\mathrm{pred}} \leftarrow \mathbf{X}^{\mathrm{pred}} \cup \{\mathbf{X}_j^{\mathrm{pred}}\}$, $\mathbf{P}^{\mathrm{pred}} \leftarrow \mathbf{P}^{\mathrm{pred}} \cup \{p_j\}$
    return $\mathbf{X}^{\mathrm{pred}}$, $\mathbf{P}^{\mathrm{pred}}$
```

twofold: 1) the base RC has the largest number of trajectories to ensure that the LSTM-based encoder–decoder is well-trained and 2) the second part of the training phase is very quick as each RCA module is a much simpler network compared to the LSTM network.

Algorithm 1 outlines the prediction process of PoPPL for the input test trajectory of a POI. The sub-LSTM mentioned in line 5 of the algorithm can be any one of the above three networks. Given m trained sub-LSTMs in stage 2, m predicted trajectories and m corresponding probabilities are returned for each test trajectory. Through this novel two-stage trajectory prediction network, our method can predict pedestrian trajectories heading toward different destination regions. This is equivalent to yielding the top k results in image/object classification problems.

E. Implementation Details of the Proposed Network

In stage 1 of PoPPL, the forward and backward LSTM layers have a fixed hidden-state dimension of 128. The combined hidden vector from these layers is then processed by a 1-D convolutional layer of kernel size 3 with 64 channels. The max-pooling layer is also 1-D with the pool size set to 2. At stage 2, a 128-D hidden state is used for all the sub-LSTMs. Given that we perform RC classification in the first stage, the cross-entropy loss is applied, whereas the mean square error (mse) (equivalent to the ADE described later) is used as the loss function for all the sub-LSTMs in the second stage. The parameters of the proposed network are trained with an RMSprop optimizer [66] and the learning rate is 0.001. During each training epoch, 20% of the training samples are randomly selected as the validation data. All the sub-LSTMs are trained for 3000 epochs. PoPPL was built using Python on Keras with a TensorFlow backend and trained with a NVIDIA GeForce GTX-1080 GPU. The networks for both stages can be trained in parallel after the source/destination clustering step. The code is available on https://github.com/xuehaouwa/poppl.

V. Experiments

A. Data Sets

1) NYGC Data Set: This data set [21] contains over 12 000 pedestrian walking routes annotated as ground truth from a one-hour surveillance video at a frame rate of 23. It is a very crowded scene in an indoor environment. There are 12 684 manually annotated pedestrians' trajectories and an average of over 100 pedestrians in each frame.

2) Edinburgh Informatics Forum Data Set: This data set consists of a set of detected targets of people walking through the Informatics Forum at the University of Edinburgh [22]. The data cover several months of observations, thus resulting in about 1000 observed trajectories for each working day (over 92 000 trajectories in total). The frame rate of the original surveillance video of the Edinburgh data set is 9 frames/s.

3) Town Centre Data Set: This data set [67] includes hundreds of human trajectories in a real-world crowded scene. In our experiments, the centers of body bounding boxes are considered as the trajectory coordinates. The video has 25 frames/s, and the resolution is 1920×1080 pixels. We preprocessed the annotated trajectories by downsampling to every fifth frame.

B. Experimental Setup

In the experiments, the lengths of the observed trajectories and the predicted trajectories are 20 frames. Trajectories shorter than 40 frames were first filtered out, thus giving us about 5000 trajectories in the NYGC data set and 25 000 trajectories in the Edinburgh data set. For the NYGC data set, we split the trajectories into a training set (80% of trajectories) and a test set (20% of trajectories). For the Edinburgh data set, we follow [13] and randomly sampled 20 000 trajectories and then 5000 trajectories to form the training set and the test set. In the RC clustering step, $\tau_{\min}$ was set to 5, which means there were at least 5% of the total number of trajectories in each RC. The threshold τ used in the second stage of our method was set to 0.01, i.e., all the sub-LSTMs whose RC probabilities were above τ were chosen to generate trajectories.

We use the Town Centre data set to test our method for different predicted trajectory lengths. Same as other data sets, 80% of trajectories were randomly sampled to form the training set and the rest of trajectories were used for testing.

C. Baselines and Metrics

In our experiments, we compare the prediction performance of the following methods.

1) *Linear Prediction:* A preliminary linear prediction method used to predict plausible trajectories with the assumption that pedestrians walk in straight paths. This method is a baseline method for trajectory prediction.
2) *LSTM:* This is the basic vanilla LSTM-based trajectory prediction method, which does not consider RC classification. This is the baseline method for the LSTM-based trajectory prediction used in this article.
3) *SF:* The SF model [9] uses factors such as preferred walking speeds, destinations, and group affinity to model pedestrians' moving patterns for trajectory prediction.
4) *Social-LSTM:* The social-LSTM model proposed by Alahi *et al.* [2] combines a social pooling layer with the traditional LSTM network to model pedestrian–pedestrian interactions in the trajectory predicting process.

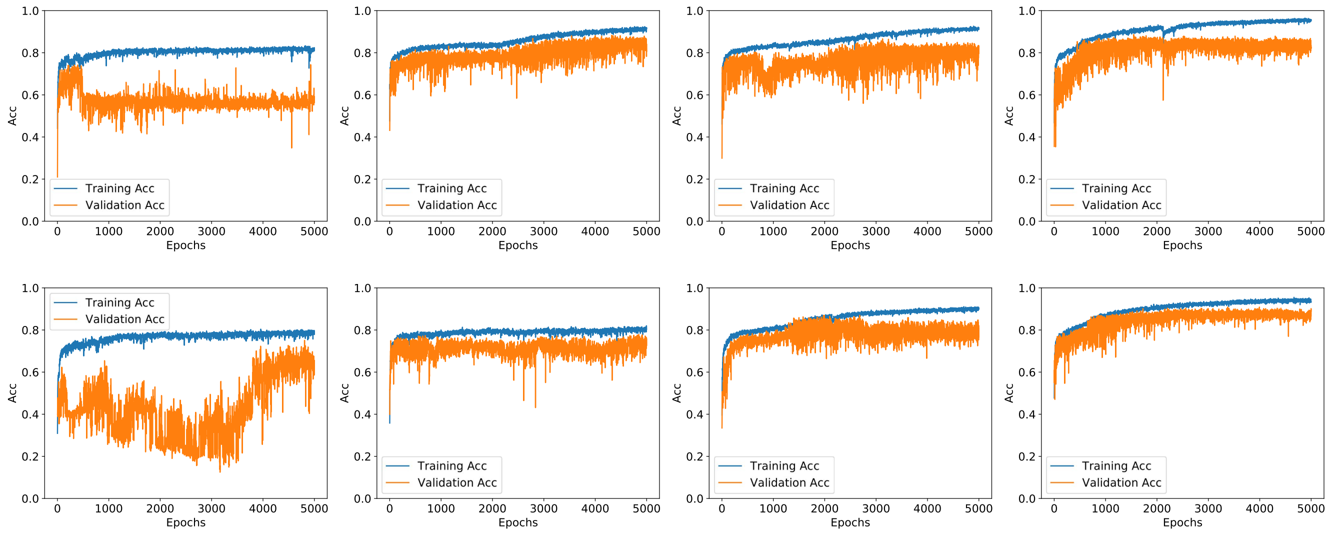


Fig. 5. Classification accuracy curves of training (blue curves) and cross-validation (orange curves) with different hyperparameter values for stage 1 of PoPPL. From left to right, each column denotes 16, 32, 64, and 128 hidden dimensions of LSTM layers, respectively. The dropout is 0.2 for the first row and 0.5 for the last row.

5) *Attention-LSTM:* An attention-based mechanism is introduced to modify the social-LSTM model by Fernando *et al.* [13], which considers the entire sequence of hidden states for both the predicted trajectory of each pedestrian and his/her neighbors.

6) *PoPPL:* Our proposed trajectory method, with PoPPL-k representing the prediction method using sub-LSTMs in the second stage with the top k probabilities (that are larger than the probability threshold τ) given by the first stage of PoPPL.

All three sub-LSTM architectures are included in the comparison. Since PoPPL-def is the default sub-LSTM network for stage 2 of our proposed method, if only the name PoPPL is mentioned, it means PoPPL-def is used.

We use the ADE [13], [65], [68] and the final displacement error (FDE) [2], [13], [65] metrics to evaluate the trajectory prediction performance of our method and other state-of-the-art approaches. The ADE is the mean Euclidean distance between all the points in the predicted and ground truth trajectories averaged over all the trajectories. The FDE is defined as the average Euclidean distance between the final points (i.e., the destination points) of the predicted trajectories and the ground truth trajectories. In summary, the ADE measures the average accuracy of the predicted trajectories, while the FDE measures their average deviation from the destinations.

D. Results

1) Hyperparameters: Three hyperparameters are crucial to the performance of the proposed network: the number of hidden units of the LSTM layers, the number of training epochs, and the dropout rate. The number of epochs indicates when to stop the training process, and a proper dropout value would help avoid overfitting. As there are two stages in PoPPL and multiple sub-LSTMs are used in the second stage, only stage 1 is used for fine-tuning the parameters through cross-validation. The training process of stage 2 follows the same parameter setting as in stage 1.

Fig. 5 shows the learning curves when BiLSTM in the first stage of PoPPL is trained under different hyperparameter settings. During each training epoch, 20% of training data are randomly selected to form the data for cross-validation. Using randomly selected data for validation may be the reason why the validation accuracy curves are noisier than the training accuracy curves. From left to right, the number of hidden units of each LSTM layer is 16, 32, 64, and 128. Compared to the first three columns, the accuracy curves for the 128 hidden units (last column) are better (more steady) and the validation accuracy is higher when dropout is 0.5 (last row). Note that the improvement from 64 to 128 hidden units is relatively small and since increasing the number of hidden units further will increase the training time, we set the hidden dimension of the LSTM layers in all our experiments to 128 as described in Section IV-E.

For the number of training epochs, we can see from Fig. 5 that the increase in accuracy of the validation set is not significant from epoch 3000 onward. Comparing the first row (dropout = 0.2) against the last row (dropout = 0.5), the accuracy curves for a higher dropout value are clearly better when the number of hidden units of the LSTM layers is large. For example, when 128 hidden units and 0.2 dropout rate are used (the last plot in the first row), the validation accuracy does not increase with increased training accuracy, which indicates overfitting on the training set. Thus, we train the network with 3000 epochs and a 0.5 dropout rate.

2) RC Classification: Each RC determines the general direction of the whole path of trajectories in that RC. The accuracy of the RC classification in stage 1 of PoPPL on the test set is given in Table II. The network was trained using 80% training trajectories and their RCs obtained from the source/destination point clustering process. The RC setting consists of two parts: the number of source/destination point clusters and the number of RCs. Under the RC setting column in Table II, $i\#j$ denotes that there are i point clusters and j RCs. Note that the number of source/destination point clusters

TABLE II
RC CLASSIFICATION ACCURACY OF STAGE 1

Dataset	*RC Setting*	Top-1 LSTM-1 / BiLSTM	Top-3 LSTM-1 / BiLSTM
NYGC	4#6	79.3% / 84.5%	97.3% / 98.8%
	5#8	72.7% / 81.9%	95.1% / 97.7%
	6#8	72.1% / 79.7%	94.4% / 97.2%
	8#9	69.2% / 74.5%	92.7% / 96.1%
Edinburgh	4#8	92.3% / 92.9%	98.8% / 99.5%
	5#9	91.6% / 92.5%	98.0% / 99.5%
	6#12	85.7% / 87.7%	96.8% / 97.8%
	8#9	85.2% / 88.1%	97.2% / 97.9%

*BiLSTM denotes that the proposed route class classification network shown in Fig. 3 is used in stage 1; LSTM-1 denotes that the forward and backward layers of BiLSTM are replaced by the traditional one-directional LSTM.

TABLE III
COMPARISON OF DIFFERENT TRAJECTORY PREDICTION METHODS. THE LOWEST AND SECOND LOWEST ERRORS ARE HIGHLIGHTED IN BLUE AND RED, RESPECTIVELY

Methods	Edinburgh ADE	Edinburgh FDE	NYGC ADE	NYGC FDE
Linear	0.803	1.526	3.218	6.989
LSTM	2.132	3.005	3.696	4.723
SF [8]	3.124*	3.909*	3.364*	5.808*
Social-LSTM [2]	1.524*	2.510*	1.990*	4.519*
Attention-LSTM [13]	0.986*	1.311*	1.096*	3.011*
PoPPL-1	0.561	0.878	1.664	3.151
PoPPL-3	0.449	0.677	1.288	2.264
PoPPL-att-1	0.408	0.725	1.513	2.974
PoPPL-att-3	0.345	0.576	1.140	2.036
PoPPL-rca-1	0.501	0.840	1.672	3.221
PoPPL-rca-3	0.396	0.617	1.304	2.314

* results extracted from [13]

and $\tau_{\min}$ govern the number of RCs as well as the value of k in the top-k prediction results. Besides our proposed bidirectional LSTM-based architecture shown in Fig. 3 (BiLSTM), we also report the accuracy of LSTM-1, a variant of BiLSTM where the bidirectional part of the network is replaced by the traditional one-directional LSTM.

The results on the Edinburgh data set have higher accuracy than those on the NYGC data set. The more complicated background scene of the NYGC data set gives more complex trajectories, thus leading to a relatively lower RC classification accuracy. For all different RC settings, using the bidirectional LSTM outperforms the traditional one-directional LSTM on both data sets. These results show that the bidirectional LSTM architecture is more suitable for RC classification in the proposed network.

The figures in Table II show that the classification accuracy drops with an increase in the number of RCs in the RC setting. This is expected as more RCs would increase the difficulties of classification. However, compared to the traditional one-directional LSTM architecture, the accuracy of using the bidirectional LSTM drops slower.

3) Quantitative Results: The quantitative results in terms of ADE and FDE are given in Table III (using meters as the unit). The best results are presented in blue, and the second-best results are presented in red. The prediction performance of all methods on the Edinburgh data set is better than that on the NYGC data set. Generally, this can be explained by different complexities of the scenes (the background scenes of these two data sets are shown in Fig. 2). It is clear that the scene from the NYGC data set is more complicated (e.g., people need to detour slightly around the information center) and more crowded (e.g., average 100 people per frame at the NYGC data set compares with average 1000 per day at the Edinburgh data set). In this situation, the basic linear prediction method performs much better on the Edinburgh data set than the NYGC data set as the Edinburgh scene was free of obstacles and people were able to walk in straight lines. Furthermore, because of the relatively simple layout of the Edinburgh scene, even the vanilla LSTM method performs better than the SF model-based method.

The results of using different sub-LSTM architectures described in Section IV-D are listed in the last six rows of Table III. For the Edinburgh data set, all variants of PoPPL give smaller prediction errors than the state-of-the-art methods. For the ADE measure on the NYGC data set, PoPPL-att-3 comes in the second place for the ADE, thus performing slightly worse than Attention-LSTM but outperforming other methods by a large margin. While the Attention-LSTM method focuses on the influence of neighbors to help to predict trajectories, PoPPL-att uses RC prediction, which pays more attention to the potential destination regions. Thus, PoPPL-att-3 gives the smallest average final destination prediction error. For the FDE measure on the NYGC data set, PoPPL-att-3 outperforms all other methods. As PoPPL-rca uses a simple RCA module to reduce the training complexity in the second stage, it is not surprising that PoPPL-rca performs relatively worse than PoPPL-att. However, PoPPL-rca performs slightly better than PoPPL on the Edinburgh data set, which suggests that PoPPL-rca still can have a good performance on simple scenes like the Edinburgh data set.

The main difference between other methods and PoPPL is the unsupervised trajectory clustering step that we introduce and apply for the RC classification network in stage 1. The experiments on these two data sets show that using trajectory clustering and RC classification to guide the prediction of trajectories can improve the prediction accuracy. Through trajectory clustering, different trajectory patterns (RC) are separated, so each sub-LSTM only needs to learn a specific and similar trajectory pattern.

4) Ablation Study: The vanilla LSTM method, which does not include classifying trajectories into RCs, is the baseline method for evaluating PoPPL. Our experiments show that PoPPL outperforms the vanilla LSTM method by a large margin on the Edinburgh and NYGC data sets (see Table III). In Fig. 6, some qualitative results from these two methods versus the ground truth for the NYGC data set are illustrated. From Fig. 6(a) and (b), we can see that the bidirectional LSTM introduced in stage 1 helps to improve the accuracy of trajectory prediction. For example, in Fig. 6(b), the prediction method without classification gives worse predictions than PoPPL.

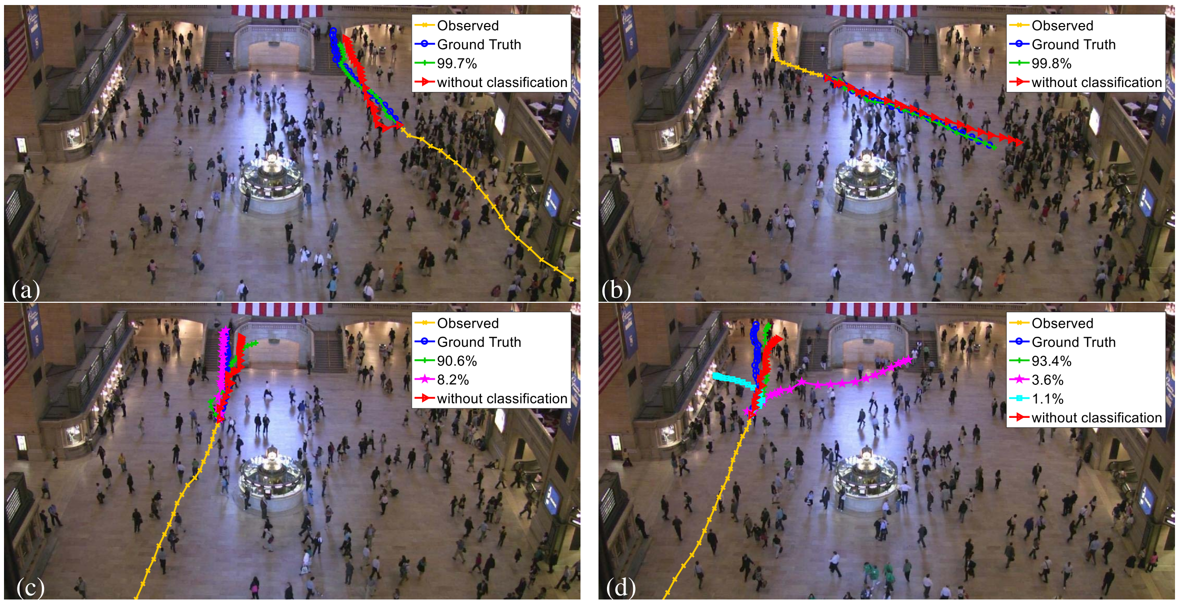


Fig. 6. (a)–(d) Illustration of a few predicted trajectories from PoPPL on the NYGC data set. The observed trajectories and ground truth are given in yellow and blue, respectively. The multiple predicted trajectories' outputs by our method are shown in green (highest probability), magenta (second), and cyan (third). We also compare our method with the vanilla LSTM method (shown in red) that does not use RC classification.

Multiple predicted trajectories with different probabilities toward different destination regions are shown in Fig. 6(c) and (d). There are two and three probabilities that are larger than the threshold τ, so our proposed method gives two and three predicted trajectories, respectively. In Fig. 6(c), compared to the predicted trajectory (the green curve with $+$ markers) with the highest probability, the second-best predicted trajectory (the magenta curve with $\star$ markers) overlaps almost perfectly with the ground truth (the blue curve with $\circ$ markers). This example shows that our proposed method is able to cover different possible intended destinations (in this example, either going upstairs or heading toward a platform) of the pedestrian in the top-k predictions. In Fig. 6(d), PoPPL gives a third predicted trajectory with a small probability (the cyan curve with $\square$ markers), while the first predicted trajectory (with a probability of 93.4%) agrees well with the ground truth. This example demonstrates that our proposed method can cover different movement patterns and the small probability where a POI might turn toward another nearby destination, such as the ticket counters in the scene.

5) Results for Different RC Settings: The trajectory prediction results on the test set for different RC settings are given in Table IV. Because the number of trajectories having the same source and destination regions in the NYGC data set is below the $\tau_{\min}$ threshold, they do not form RCs. Hence, for the case where there are four source/destination point clusters, there are only three other potential destination regions that a pedestrian can go to. The row 4#6 of NYGC, therefore, has no PoPPL-4 and PoPPL-5 prediction results.

In general, given N source/destination point clusters in a scene, there are no more than N destination regions to which a pedestrian can go from any specific source region. If the number of RCs is fewer than $N(N+1)/2$, then this number of destination regions may be less than N. Thus, for the row 5#8 for NYGC, due to the small number of RCs (8 in this case), the number of destination regions are four instead of five, and only the PoPPL-4 accuracy can be computed. Similarly for other rows in Table IV, only the PoPPL-k accuracies with the highest k are reported if the PoPPL-5 accuracy cannot be computed.

TABLE IV
PREDICTION ERRORS (IN METERS) FOR DIFFERENT RC SETTINGS ON THE NYGC AND THE EDINBURGH DATA SETS

Dataset	*RC*	PoPPL-1		PoPPL-3		PoPPL-5	
	Setting	ADE	FDE	ADE	FDE	ADE	FDE
NYGC	4#6	**1.664**	**3.151**	1.288	2.264	-	-
	5#8	1.812	3.555	1.367	2.324	1.289[*]	2.119[*]
	6#8	1.745	3.487	1.281	2.215	1.195	**1.967**
	8#9	1.681	3.311	**1.249**	**2.208**	**1.084**	2.036
Edin-burgh	4#8	0.561	**0.878**	0.449	0.677	0.424[*]	0.637[*]
	5#9	0.597	0.938	0.471	0.726	0.442	0.663
	6#12	**0.543**	0.895	**0.407**	**0.668**	**0.396**	**0.621**
	8#9	0.621	0.990	0.476	0.739	0.443	0.622

[*] PoPPL-4 accuracy only

Comparing the prediction errors (separately for ADEs and FDEs) shown in Table IV, as expected, all the RC settings have their PoPPL-k values decrease with an increase in k. However, if the RC setting has more RCs, then both the ADE and FDE values appear to drop at a faster rate. Hence, even though it has been shown in Section V-D2 that fewer RCs favor the RC classification in stage 1, having more RCs seems to be more advantageous when multiple predictions need to be generated. Consider the classification accuracy in stage 1 for BiLSTM on the NYGC data set shown in Table II. For the two RC settings 4#6 and 8#9, increasing the number of RCs from 6 to 9 decreases the top-3 accuracy by 2.7% (from 98.8% to 96.1%). However, as shown in Table IV, the three extra RCs make their PoPPL-3 ADE value drop from 1.288 to 1.249.

TABLE V

PREDICTION ERRORS (IN METERS) ON THE EDINBURGH DATA SET FOR DIFFERENT NUMBERS OF RCs FOR SIX SOURCE/DESTINATION REGIONS

# Route	PoPPL-1		PoPPL-3		PoPPL-5	
Classes	ADE	FDE	ADE	FDE	ADE	FDE
4	0.694	1.146	0.572	0.905	0.541*	0.859*
6	0.689	1.093	0.507	0.878	0.493	0.760
8	0.623	1.002	0.486	0.821	0.475	0.697
10	0.595	0.919	0.474	0.764	0.436	0.684
12	**0.543**	**0.895**	**0.407**	**0.668**	**0.396**	**0.621**

* PoPPL-4 accuracy only

TABLE VI

TRAINING TIMES OF PoPPL, PoPPL-ATT, PoPPL-RCA, AND THE VANILLA LSTM METHOD

Methods	Dataset	Training Time (minutes)
LSTM	NYGC	71
PoPPL (total)		105 ± 4
PoPPL-att (total)		126 ± 10
PoPPL-rca (total)		65 ± 8
PoPPL (stage 1)		46 ± 6
LSTM	Edinburgh	364
PoPPL (total)		508 ± 34
PoPPL-att (total)		602 ± 48
PoPPL-rca (total)		329 ± 41
PoPPL (stage 1)		226 ± 13

To further investigate the relationship between the number of RCs and the prediction performance, we use the six-point cluster setting (i.e., the RC setting 6#12 in Table IV) for the Edinburgh data set because this RC setting has the largest RC number. All the RCs are first sorted in the decreasing order of the number of training trajectories. Four PoPPL models are then, respectively, trained using the top 4, 6, 8, and 10 RCs. The PoPPL-1, PoPPL-3, and PoPPL-5 ADE and FDE values of the four models above, together with the 12 RC model from Table IV, are shown in Table V. As expected, both ADE and FDE decrease with more RCs added to train the network and using the 12 RC setting outperforms other settings.

6) Computation Times: Since the vanilla LSTM is the simplest LSTM-based trajectory prediction method, it serves as a good baseline method for speed comparison. We train five models of PoPPL, PoPPL-att, and PoPPL-rca for five different N (the number of source/destination regions) values (with $N = 4, \dots, 8$) to compare the variation of their processing time with that of the vanilla LSTM. For all methods, we use 3000 epochs and the same GTX-1080 GPU desktop. Table VI shows their ranges of processing times in minutes in the form of $a \pm b$, where each a is the average of the lowest and highest processing times of the associated model and b is half of their difference. As both stages 1 and 2 of PoPPL require training, we report the total training time and the training time for stage 1 alone.

As expected, the training time is related to the size of the data set. Given that the size of Edinburgh is five times of NYGC, its training time is around five times as much. Compared to the vanilla LSTM method, the extra training times of the three PoPPL models are mostly due to the extra RC classification part in stage 1. In stage 2 of PoPPL, the training trajectories are divided into the sub-LSTMs. As RCs having fewer than $\tau_{\min}\%$ of the total number of trajectories are discarded, the trajectories of all the sub-LSTMs put together may be fewer than the number of trajectories passed to the vanilla LSTM. Because of that, the total training time of these sub-LSTMs is shorter than the time required to train a vanilla LSTM on all the trajectories.

Using different N values results in different numbers of RCs, and thus, different numbers of sub-LSTMs need to be trained. However, for both data sets, PoPPL, PoPPL-att, and PoPPL-rca all have their b/a values equal to 13% or less, thus indicating that their training times for different N values are very similar. These similar training times are also due to the total numbers of training trajectories being similar to these models.

Comparing the training times of PoPPL with its variants, PoPPL-att and PoPPL-rca, we observe that the extra attention mechanism in PoPPL-att results in the longest training time. As expected, the training time for PoPPL-rca is the shortest since only the base RC needs to be trained and the RCA modules only demand little extra training time.

As for the prediction time, both PoPPL and vanilla LSTM methods are about the same. Unlike high-dimensional image data, the input data of the trajectory prediction network are 2-D coordinates at each time step, so the prediction of each observed trajectory is very fast for both methods.

7) Comparison With Manual Scene Partitioning: In [65], both the NYGC and Edinburgh scenes were manually partitioned into four regions and trajectories were clustered into six RCs (i.e., the RC setting is 4#6) with the RCs $R_{i,i}$ ignored instead of using $\tau_{\min}$ to rule out RCs as done in PoPPL. This manual process has two major drawbacks: 1) it is not optimal for a deep learning-based prediction method and 2) it is subjective and time-consuming to manually label the source/destination points.

TABLE VII

PREDICTION ERRORS (IN METERS) OF THE TOP-1 RESULTS FROM PoPPL, PoPPL-RCA, AND PoPPL-ATT VERSUS THE MANUAL SCENE PARTITIONING METHOD FOR FOUR SOURCE/DESTINATION POINT CLUSTERS

	Edinburgh		NYGC	
	ADE	FDE	ADE	FDE
Bi-Prediction [65]	0.648	1.027	1.686	3.037
PoPPL	0.561	0.878	1.664	3.151
PoPPL-att	**0.408**	**0.725**	**1.513**	**2.974**
PoPPL-rca	0.501	0.840	1.672	3.221

Table VII shows the top-1 prediction results of the manual source/destination region partitioning process (Bi-Prediction [65]) versus the PoPPL models that use automatic RC clustering in this article: PoPPL, PoPPL-att, and PoPPL-rca. For these three PoPPLbased methods, we also set the number of source/destination point clusters to 4. With the help of introduced attention mechanism, PoPPL-att stands out and outperforms others on both data sets. For the Edinburgh data set, all three models with automatic trajectory clustering perform better than the manual method. For the NYGC data set, PoPPL and Bi-Prediction have

TABLE VIII

PREDICTION ERRORS (IN PIXELS) ON THE TOWN CENTRE DATA SET FOR SMALL AND LARGE PREDICTION HORIZONS. THE TOP TWO PERFORMERS ARE SHOWN IN BLUE AND RED

T_{ph} (secs)	LSTM ADE/FDE	Social-LSTM ADE/FDE	PoPPL-1 ADE/FDE	PoPPL-3 ADE/FDE
0.8	46.99/52.68	33.62/36.62	25.98/32.31	24.05/28.62
3.2	105.41/127.44	93.53/ 115.22	69.02/79.45	55.85/61.71

similar performance and the simplest PoPPL-rca has worse performance than PoPPL and PoPPL-att. Compared to the NYGC data set, the Edinburgh data set is relatively simple (having almost no obstacles) and has a larger number of trajectories. As a result, PoPPL-rca performs better in this data set. The above comparison with the manual scene partitioning demonstrates the resilience of our proposed method. We can use PoPPL-att on more complex scenes to yield better predictions and PoPPL-rca on simpler scenes while saving computation time.

8) Results on Different Prediction Horizons: The term prediction horizon, introduced by Vemula *et al.* [6], is commonly referred to as the temporal length of the predicted trajectories. Prediction horizon, which we denote by T_{ph}, can be expressed in the number of frames or a time value (e.g., in seconds) if the frame rate is known. The prediction horizon T_{ph} is the length of the predicted trajectories at the testing phase and is not always the same as the T_{pred} value set up in the training phase.

In this part of our experiments, we use the Town Centre data set to evaluate the prediction performance of our proposed method for different T_{ph} values. Same as before, four source/destination point clusters are obtained from k-means clustering, thus yielding four RCs after applying the $\tau_{\min}$ threshold. These RCs are: from bottom-left to top-right corners of the scene; top right to bottom right; bottom left to top middle; and bottom right to top middle (recall that trajectories in opposite directions are in the same RC). The same hyperparameter values as previous experiments are used for this data set.

The prediction errors (in pixel unit) of PoPPL versus those of the vanilla LSTM and social-LSTM methods on small ($T_{\text{ph}} = 0.8$ s) and large ($T_{\text{ph}} = 3.2$ s) prediction horizons are shown in Table VIII. In both cases, $T_{\text{obs}} = 1.6$ s. In Table VIII, we consider the case where $T_{\text{pred}} = T_{\text{ph}}$. When $T_{\text{obs}} > T_{\text{ph}}$, the performances from social-LSTM and PoPPL are about the same. Since the RCs of the observed trajectories have been predicted in stage 1 of PoPPL, the more specifically-trained sub-LSTMs help to guide PoPPL to steer the test trajectories toward the destination regions, which are useful for predicting long trajectories. Hence, when $T_{\text{ph}} > T_{\text{obs}}$, the ADE and FDE of PoPPL do not increase as fast as in other methods. The benefit of the RC classification stage of PoPPL is demonstrated in these experiments for predicting long trajectories. Two examples of the prediction results for $T_{\text{ph}} = 3.2$ s are illustrated in Fig. 7. The background image in each example corresponds to the frame $t = T_{\text{obs}}$. Compared to the vanilla LSTM and social-LSTM methods, predicted trajectories from PoPPL are closer to the ground truth trajectories.

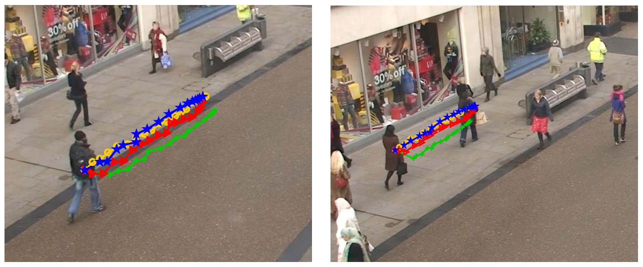


Fig. 7. Two examples of the prediction results on the Town Centre data set in the case of large T_{ph} (3.2 s). Color codes: blue: PoPPL-1; red: social-LSTM; green: vanilla LSTM; and yellow: ground truth.

VI. CONCLUSION

We have presented a novel method for predicting pedestrians' trajectories in crowded scenes. RCs are used to capture movement patterns from a source region to a destination region. Using the RCs obtained from clustering, we have trained a bidirectional LSTM-based classification network in the first stage to classify trajectories into RCs to which they likely belong. In the second stage of our algorithm, one of the three proposed sub-LSTM networks can be used for trajectory prediction for the RCs. We have compared our method with other state-of-the-art methods and the results show that incorporating RC information improves the accuracy of trajectory prediction. Furthermore, PoPPL generates better predicted trajectories for the case when the prediction length of the trajectories is long. We have also demonstrated in the article that an efficient source/destination point clustering for RCs can be used to eliminate the manual source/destination region partitioning without compromising the prediction accuracy. As multiple predicted trajectories with corresponding probabilities are generated by the algorithm, our algorithm can be used to identify the busy areas of the scene where the predicted trajectories have high probabilities and isolate possible anomalous events when the predicted trajectories have very low probabilities.

In the future, we will focus on deploying our trajectory prediction method to more complex real-world applications. One interesting future research direction is to use trajectory prediction to develop an anomaly detection and prevention system in surveillance. By using more RCs, pedestrians' movement patterns can be captured at a finer scale, which can, in turn, enhance the performance of anomaly detection.

REFERENCES

[1] S. Kim *et al.*, "BRVO: Predicting pedestrian trajectories using velocity-space reasoning," *Int. J. Robot. Res.*, vol. 34, no. 2, pp. 201–217, Dec. 2014.
[2] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *Proc. CVPR*, Jun. 2016, pp. 961–971.
[3] H. Su, Y. Dong, J. Zhu, H. Ling, and B. Zhang, "Crowd scene understanding with coherent recurrent neural networks," in *Proc. IJCAI*, 2016, pp. 3469–3476.

[4] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. S. Torr, and M. Chandraker, "DESIRE: Distant future prediction in dynamic scenes with interacting agents," in *Proc. CVPR*, Jul. 2017, pp. 336–345.

[5] H. Su, J. Zhu, Y. Dong, and B. Zhang, "Forecast the plausible paths in crowd scenes," in *Proc. IJCAI*, Aug. 2017, pp. 2772–2778.

[6] A. Vemula, K. Muelling, and J. Oh, "Modeling cooperative navigation in dense human crowds," in *Proc. ICRA*, May 2017, pp. 1685–1692.

[7] H. Xue, D. Q. Huynh, and M. Reynolds, "SS-LSTM: A hierarchical LSTM model for pedestrian trajectory prediction," in *Proc. WACV*, Mar. 2018, pp. 1186–1194.

[8] D. Helbing and P. Molnár, "Social force model for pedestrian dynamics," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 51, no. 5, pp. 4282–4286, May 1995.

[9] K. Yamaguchi, A. C. Berg, L. E. Ortiz, and T. L. Berg, "Who are you with and where are you going?" in *Proc. CVPR*, Jun. 2011, pp. 1345–1352.

[10] D. Xie, S. Todorovic, and S. C. Zhu, "Learning and inferring 'dark matter' and predicting human intents and trajectories in videos," in *Proc. ICCV*, Dec. 2013, pp. 2224–2231.

[11] I. Karamouzas, B. Skinner, and S. J. Guy, "Universal power law governing pedestrian interactions," *Phys. Rev. Lett.*, vol. 113, no. 23, Dec. 2014, Art. no. 238701.

[12] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. W. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in *Proc. IJCAI*, Aug. 2017, pp. 2627–2633.

[13] T. Fernando, S. Denman, S. Sridharan, and C. Fookes, "Soft+hardwired attention: An LSTM framework for human trajectory prediction and abnormal event detection," *Neural Netw.*, vol. 108, pp. 466–478, Dec. 2018.

[14] A. Vemula, K. Muelling, and J. Oh, "Social attention: Modeling attention in human crowds," 2017, *arXiv:1710.04689*. [Online]. Available: http://arxiv.org/abs/1710.04689

[15] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. ICLR*, 2015, pp. 1–15.

[16] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. EMNLP*, 2015, pp. 1412–1421.

[17] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa, "Visual domain adaptation: A survey of recent advances," *IEEE Signal Process. Mag.*, vol. 32, no. 3, pp. 53–69, May 2015.

[18] Y. Du, W. Jin, W. Wei, Y. Hu, and W. Geng, "Surface EMG-based inter-session gesture recognition enhanced by deep domain adaptation," *Sensors*, vol. 17, no. 3, p. 458, Feb. 2017.

[19] A. Rosenfeld and J. K. Tsotsos, "Incremental learning through deep adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 42, no. 3, pp. 651–663, Mar. 2020.

[20] Y. Zou, Z. Yu, B. V. Kumar, and J. Wang, "Unsupervised domain adaptation for semantic segmentation via class-balanced self-training," in *Proc. ECCV*, Sep. 2018, pp. 297–313.

[21] S. Yi, H. Li, and X. Wang, "Understanding pedestrian behaviors from stationary crowd groups," in *Proc. CVPR*, Jun. 2015, pp. 3488–3496.

[22] B. Majecka, "Statistical models of pedestrian behaviour in the forum," M.S. thesis, School Informat., Univ. Edinburgh, Scotland, U.K., 2009.

[23] B. Morris and M. Trivedi, "Learning trajectory patterns by clustering: Experimental studies and comparative evaluation," in *Proc. CVPR*, Jun. 2009, pp. 312–319.

[24] G. Antonini and J. P. Thiran, "Counting pedestrians in video sequences using trajectory clustering," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 16, no. 8, pp. 1008–1020, Aug. 2006.

[25] T. Zhang, H. Lu, and S. Z. Li, "Learning semantic scene models by object classification and trajectory clustering," in *Proc. CVPR*, Jun. 2009, pp. 1940–1947.

[26] X. Wang, K. Tieu, and E. Grimson, "Learning semantic scene models by trajectory analysis," in *Proc. ECCV*, 2006, pp. 110–123.

[27] W. Ge, R. T. Collins, and R. B. Ruback, "Vision-based analysis of small groups in pedestrian crowds," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 5, pp. 1003–1016, May 2012.

[28] B. T. Morris and M. M. Trivedi, "Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 11, pp. 2287–2301, Nov. 2011.

[29] H.-Y. Hu, Z.-W. Qu, and Z.-H. Li, "Multi-level trajectory learning for traffic behavior detection and analysis," *J. Chin. Inst. Eng.*, vol. 37, no. 8, pp. 995–1006, May 2014.

[30] C. Piciarelli and G. L. Foresti, "On-line trajectory clustering for anomalous events detection," *Pattern Recognit. Lett.*, vol. 27, no. 15, pp. 1835–1842, Nov. 2006.

[31] D. Kumar, J. C. Bezdek, S. Rajasegarar, C. Leckie, and M. Palaniswami, "A visual-numeric approach to clustering and anomaly detection for trajectory data," *Vis. Comput.*, vol. 33, no. 3, pp. 265–281, Dec. 2015.

[32] C. Piciarelli, C. Micheloni, and G. L. Foresti, "Trajectory-based anomalous event detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 11, pp. 1544–1554, Nov. 2008.

[33] R. Agrawal, C. Faloutsos, and A. Swami, "Efficient similarity search in sequence databases," in *Foundations of Data Organization and Algorithms*. Berlin, Germany: Springer, 1993, pp. 69–84.

[34] L. R. Rabiner and B.-H. Juang, *Fundamentals of Speech Recognition*, 1st ed. Upper Saddle River, NJ, USA: Prentice-Hall, 1993.

[35] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Proc. 18th Int. Conf. Data Eng.*, Mar. 2002, pp. 673–684.

[36] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proc. ACM SIGMOD Int. Conf. Manage.*, 2005, pp. 491–502.

[37] D. F. Silva and G. E. A. P. A. Batista, "Speeding up all-pairwise dynamic time warping matrix calculation," in *Proc. SIAM Int. Conf. Data Mining*, Aug. 2016, pp. 837–845.

[38] T. Pratzlich, J. Driedger, and M. Muller, "Memory-restricted multiscale dynamic time warping," in *Proc. ICASSP*, Mar. 2016, pp. 569–573.

[39] S. Atev, G. Miller, and N. P. Papanikolopoulos, "Clustering of vehicle trajectories," *IEEE Trans. Intell. Transp. Syst.*, vol. 11, no. 3, pp. 647–657, Sep. 2010.

[40] D. Ellis, E. Sommerlade, and I. Reid, "Modelling pedestrian trajectory patterns with Gaussian processes," in *Proc. ICCV Workshops*, Sep. 2009, pp. 1229–1234.

[41] P. Trautman and A. Krause, "Unfreezing the robot: Navigation in dense, interacting crowds," in *Proc. IROS*, Oct. 2010, pp. 797–803.

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

[43] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," 2014, *arXiv:1412.3555*. [Online]. Available: http://arxiv.org/abs/1412.3555

[44] T. Lei, Y. Zhang, S. I. Wang, H. Dai, and Y. Artzi, "Simple recurrent units for highly parallelizable recurrence," 2017, *arXiv:1709.02755*. [Online]. Available: http://arxiv.org/abs/1709.02755

[45] J. Donahue *et al.*, "Long-term recurrent convolutional networks for visual recognition and description," in *Proc. CVPR*, 2015, pp. 2625–2634.

[46] A. Graves, A. R. Mohamed, and G. Hinton, "Towards end-to-end speech recognition with recurrent neural networks," in *Proc. ICML*, vol. 14. 2014, pp. 1764–1772.

[47] J. S. Ren *et al.*, "Look, listen and learn—A multimodal LSTM for speaker identification.," in *Proc. AAAI*, 2016, pp. 3581–3587.

[48] J.-T. Chien and Y.-C. Ku, "Bayesian recurrent neural network for language modeling," *IEEE Trans. Neural Netw. Learn. Systems*, vol. 27, no. 2, pp. 361–374, Feb. 2016.

[49] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. NIPS*, 2014, pp. 3104–3112.

[50] W. Zhu *et al.*, "Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks," in *Proc. AAAI*, 2016, pp. 3697–3703.

[51] J. Liu, G. Wang, P. Hu, L.-Y. Duan, and A. C. Kot, "Global context-aware attention LSTM networks for 3D action recognition," in *Proc. CVPR*, Jul. 2017, pp. 1647–1656.

[52] J. Liu, A. Shahroudy, D. Xu, A. C. Kot, and G. Wang, "Skeleton-based action recognition using spatio-temporal LSTM network with trust gates," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 40, no. 12, pp. 3007–3021, Dec. 2018.

[53] X. Shu, J. Tang, G.-J. Qi, W. Liu, and J. Yang, "Hierarchical long short-term concurrent memory for human interaction recognition," 2018, *arXiv:1811.00270*. [Online]. Available: http://arxiv.org/abs/1811.00270

[54] A. Karpathy and F.-F. Li, "Deep visual-semantic alignments for generating image descriptions," in *Proc. CVPR*, 2015, pp. 3128–3137.

[55] M. Chen, G. Ding, S. Zhao, H. Chen, Q. Liu, and J. Han, "Reference based LSTM for image captioning," in *Proc. AAAI*, 2017, pp. 3981–3987.

[56] A. Graves, "Generating sequences with recurrent neural networks," 2013, *arXiv:1308.0850*. [Online]. Available: http://arxiv.org/abs/1308.0850

[57] Y. Endo, K. Nishida, H. Toda, and H. Sawada, "Predicting destinations from partial trajectories using recurrent neural network," in *Proc. PAKDD*, 2017, pp. 160–172.

[58] X. Song, H. Kanasugi, and R. Shibasaki, "DeepTransport: Prediction and simulation of human mobility and transportation mode at a citywide level," in *Proc. IJCAI*, 2016, pp. 2618–2624.

[59] Q. Liu, S. Wu, L. Wang, and T. Tan, "Predicting the next location: A recurrent model with spatial and temporal contexts," in *Proc. AAAI*, 2016, pp. 194–200.

[60] L. Sun, Z. Yan, S. M. Mellado, M. Hanheide, and T. Duckett, "3DOF pedestrian trajectory prediction learned from long-term autonomous mobile robot deployment data," 2017, *arXiv:1710.00126*. [Online]. Available: http://arxiv.org/abs/1710.00126

[61] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proc. ICASSP*, May 2013, pp. 6645–6649.

[62] M. Sundermeyer, T. Alkhouli, J. Wuebker, and H. Ney, "Translation modeling with bidirectional recurrent neural networks," in *Proc. EMNLP*, 2014, pp. 14–25.

[63] P. Doetsch, A. Zeyer, and H. Ney, "Bidirectional decoder networks for attention-based end-to-end offline handwriting recognition," in *Proc. ICFHR*, Oct. 2016, pp. 361–366.

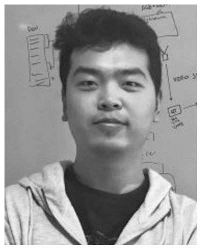
[64] A. Graves, N. Jaitly, and A.-R. Mohamed, "Hybrid speech recognition with deep bidirectional LSTM," in *Proc. IEEE Workshop Autom. Speech Recognit. Understand.*, Dec. 2013, pp. 273–278.

[65] H. Xue, D. Q. Huynh, and M. Reynolds, "Bi-prediction: Pedestrian trajectory prediction based on bidirectional LSTM classification," in *Proc. DICTA*, Nov. 2017, pp. 307–314.

[66] Y. Dauphin, H. de Vries, and Y. Bengio, "Equilibrated adaptive learning rates for non-convex optimization," in *Proc. NIPS*, 2015, pp. 1504–1512.

[67] B. Benfold and I. Reid, "Stable multi-target tracking in real-time surveillance video," in *Proc. CVPR*, Jun. 2011, pp. 3457–3464.

[68] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *Proc. ICCV*, Sep. 2009, pp. 261–268.



Hao Xue (Student Member, IEEE) received the B.S. and M.S. degrees from the Harbin Institute of Technology, Harbin, China, in 2014 and 2016, respectively. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Software Engineering, The University of Western Australia, Perth, WA, Australia.

His research is mainly on the aspect of pedestrian trajectory prediction.

Du Q. Huynh (Senior Member, IEEE) received the Ph.D. degree in computer vision from The University of Western Australia (UWA), Perth, WA, Australia, in 1994.

Since 1994, she has been with the Cooperative Research Centre for Sensor Signal and Information Processing, Adelaide, SA, Australia, and Murdoch University, Perth. She is currently an Associate Professor with the Department of Computer Science and Software Engineering, UWA. She has previously researched shape from motion, multiple view geometry, and 3-D reconstruction. Her current research interests include visual target tracking, video image processing, machine learning, and pattern recognition.

Mark Reynolds (Member, IEEE) received the B.Sc. degree (Hons.) in pure mathematics and statistics from The University of Western Australia (UWA), Perth, WA, Australia, in 1984, the Ph.D. degree in computing from the Imperial College of Science and Technology, University of London, London, U.K., 1989, and the Diploma of Education degree from UWA in 1989.

He is currently a Professor and the Head of the School of Physics, Mathematics and Computing, UWA. His current research interests include artificial intelligence, optimization of schedules and real-time systems, optimization of electrical power distribution networks, machine learning, and data analytics.