# Encoding Crowd Interaction with Deep Neural Network for Pedestrian Trajectory Prediction

Yanyu Xu *
ShanghaiTech University
xuyy2@shanghaitech.edu.cn

Zhixin Piao*
ShanghaiTech University
piaozhx@shanghaitech.edu.cn

Shenghua Gao †
ShanghaiTech University
gaoshh@shanghaitech.edu.cn

## Abstract

*Pedestrian trajectory prediction is a challenging task because of the complex nature of humans. In this paper, we tackle the problem within a deep learning framework by considering motion information of each pedestrian and its interaction with the crowd. Specifically, motivated by the residual learning in deep learning, we propose to predict displacement between neighboring frames for each pedestrian sequentially. To predict such displacement, we design a crowd interaction deep neural network (CIDNN) which considers the different importance of different pedestrians for the displacement prediction of a target pedestrian. Specifically, we use an LSTM to model motion information for all pedestrians and use a multi-layer perceptron to map the location of each pedestrian to a high dimensional feature space where the inner product between features is used as a measurement for the spatial affinity between two pedestrians. Then we weight the motion features of all pedestrians based on their spatial affinity to the target pedestrian for location displacement prediction. Extensive experiments on publicly available datasets validate the effectiveness of our method for trajectory prediction.*

## 1. Introduction

Pedestrian trajectory prediction aims to predict a continuous set of location coordinates of a pedestrian in future based on its history path, and it is an important task in computer vision because of its potential applications in behavior prediction [24] [4], traffic flow segmentation [22], crowd motion analysis [31], crowd counting and segmentation [27], abnormal detection [16], *etc.* . Tremendous efforts have been made to solve this problem [3] [10] [29] [31]. However, due to the complex nature of pedestrians, it remains a challenging problem. In practice, to make the problem tractable, some work has attempted to model the task by only considering a few factors related to pedestrian trajectory, including decision making process of individuals [10], interactions between the different pedestrians [25], and historical motion statistics of each pedestrian [26].

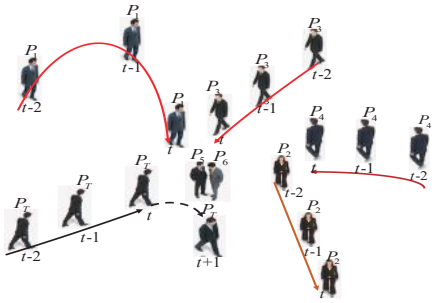


Figure 1. CIDNN motivation illustration. The motion for the target pedestrian ($P_T$) from time $t$ to $t$+1 depends on its motion, and its spatial affinity to other pedestrians($P_5$ and $P_6$) at time $t$, and other pedestrians' motion. Though its distance to pedestrian $P_1$ is far, but $P_1$ moves fast, so it also influences the movement of $P_T$. So trajectory prediction of $P_T$ should consider more pedestrians other than its neighbors with a fixed distance, and different pedestrians also have different level of influence on the target pedestrian.

In light of the success of deep learning in computer vision, it has also been introduced to pedestrian trajectory prediction, of which Behavior Convolutional Neural Network (Behavior CNN) [25] and Social Long-Short Term Memory (Social LSTM) [1] are two representative ones. Behavior CNN represents historical trajectories of all pedestrians with a position displacement map in the image space, and then a CNN is adapted to associate each pedestrian with its neighbors for future trajectory prediction. But such method cannot model the potential interactions between pedestrians in a more distant future. For example, as shown in Fig. 1 a pedestrian walking very fast in a far distance may also influence the walking trajectory of the target pedestrian, or if a group of pedestrians are walking towards the target or they are standing in the target pedestrian's walking direction, even they are far from the target pedestrian, he/she may change his/her walking direction in advance to keep away

*Equal contributions
†Corresponding author

from these people in advance. To prevent this, Social LSTM [1] is proposed. It designs a social pooling layer to capture dependencies between multiple pedestrians and interactions that could occur in a more distant future, thus achieves better performance. However, such a social pooling does not differentiate the effect of neighboring pedestrians based on their spatial positions and their motion information.

In this paper, we propose a Crowd Interaction Deep Neural Network framework (CIDNN) to sequentially predict the coordinate displacement between two frames for each pedestrian. We assume the movement of the target pedestrian depends on its motion information(speed, acceleration), other pedestrians motion information, as well as the spatial affinity between the target and all the rest pedestrians, where the spatial affinity measures the level of influence of the rest pedestrians to the target pedestrian. To model the motion of each pedestrian, an LSTM model is adapted whose input is the coordinate sequence at different moments of the pedestrian. To measure the spatial affinity of each pedestrian to the target pedestrian at a given moment, we feed the coordinates of a pedestrian into a multi-layer perceptron, and use the inner product between the coordinate feature of the pedestrian and that of the target to measure spatial affinity. Then we module the interactions between the target and all pedestrians including itself as the product between their spatial affinity and the motion feature of the corresponding pedestrian, and feed the interaction features into another multi-layer perceptron for coordinate displacement prediction of the target in next frame. We consider all pedestrians in the scene as well as their spatial affinity for trajectory prediction, thus as demonstrated in Table 1, our method outperforms both LSTM [1] and Behavior CNN[25]. Further, different from Social LSTM and Behavior CNN that directly predict the coordinates, we propose to predict the location displacement between between next and current frame, which further validates the effectiveness of residual learning in computer vision [8][9].

The contributions of our paper can be summarized as follows: Firstly, we propose a CIDNN architecture for trajectory prediction, which considers all pedestrians in a scene for trajectory prediction. Our CIDNN has three features: i)an LSTM based motion encoding strategy; ii) location based spatial affinity measurement; and iii) coordinate displacement based trajectory prediction. We propose to use location based spatial affinity measurement module, which experimentally shown its good performance than distance based spatial affinity. Ours takes coordinates as input to enrich the number of training samples and facilitate the network training. Consequently, our network architecture of CIDNN is simple and can be paralleled easily. Therefore, our trajectory prediction is more effective and efficient than existing methods; Finally, extensive experiments validate the effectiveness of our model for trajectory prediction.

## 2. Related Work

### 2.1. Hand-crafted Features Based Trajectory Prediction

Social force models and topic models are commonly used for hand-crafted features based trajectory prediction. Social force models learn the motion patterns based on the interactions between pedestrians. It is first proposed to model the attractive and repulsive forces in [10]. Later Mehran *et al.* propose to use social force model to learn the interaction forces between people in [16]. Antonini *et al.* [2] propose a discrete choice framework to predict pedestrian's next step under assumption that the destination and the route are known. Different from social force models, topics models [22] [11] [6] model the motion pattern based on spatial and temporal information. Further, Trajectory clustering [13] [17] [21] are also used for crowd flow estimation by clustering different trajectories into different classes. However, all these methods are based hand-crafted features, which limits the performance of trajectory prediction.

### 2.2. Deep Neural Networks Based Trajectory Prediction

Deep learning based methods have been introduced for pedestrian trajectory prediction [1] [25] [7] in light of its good performance for many computer vision tasks [19][12]. Specifically, Behaviour-CNN [25] employs a 2D map to encode the history walking path and use a CNN to model the interactions between different pedestrians, yet it doesn't consider the effect of pedestrians in a more distant future. Social LSTM [1] for human trajectory prediction design a Social Pooling layer to capture dependencies between multiple correlated sequences and interactions that could occur in a more distant future, but it doesn't consider the different importance of different pedestrians. In [14] Lee *et al.* employ RNN to capture past motion histories, the semantic scene context and interactions among multiple agents for trajectory predictions in dynamic scenes. In [7], Su *et al.* propose to deploy long short-term memory (LSTM) networks with social-aware recurrent Gaussian processes to model the complex transitions and uncertainties of the crowd and achieves good performance for trajectory prediction. But it also only considers the neighboring pedestrians and does not treat them differently. As aforementioned, some pedestrians in a far distance but with a fast movement speed may also influence the target pedestrian's trajectory in next moment, and different pedestrians have a different influence level on the target pedestrians trajectory. In this paper, we propose to take both factors into our consideration for trajectory prediction.

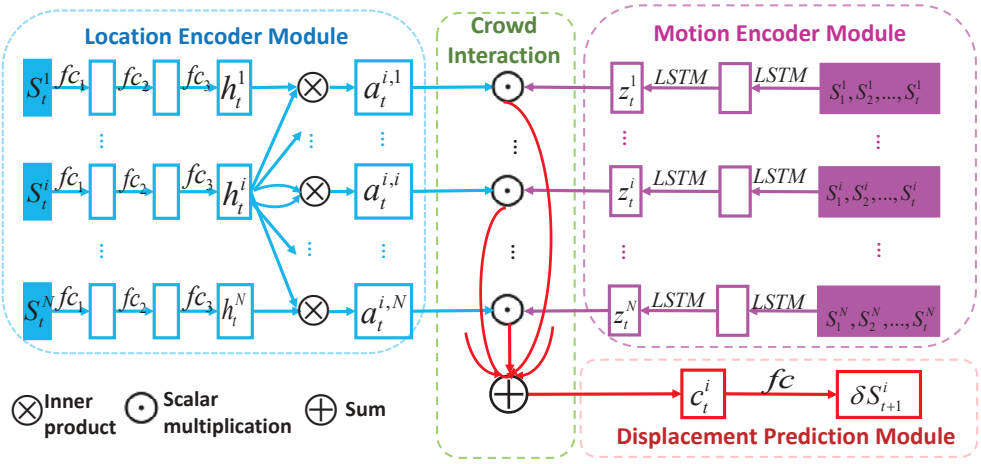



Figure 2. The architecture of crowd interaction deep neural network (CIDNN).

# 3. Method

## 3.1. The Formulation for Pedestrian Trajectory Prediction

Assume that there are $N$ pedestrians $p_1, \ldots, p_N$ , and $t$ is current time stamp (frame). The spatial location (coordinate) of the $i^{th}$ pedestrian $p_i (i \in [1, N])$ at time $t$ is denoted as $S_t^i = [x_t^i, y_t^i]$, where $x_t^i \in [1, X]$, $y_t^i \in [1, Y]$, and $[X, Y]$ is the spatial resolution of video frames.

Given the spatial coordinates $S_{1:t}$ of each pedestrian from beginning to time $t$, trajectory prediction aims at predicting the coordinates in the future time period from $t+1$ to $t+T$, i.e., $S_{t+1:t+T}$. Different from previous work [25] which predicts all the coordinates in all these frames simultaneously, we sequentially predict the coordinates in each future frame. Further, much previous work shows that the residual learning or displacement prediction is easier for image classification [8], face alignment [28], as well as pose estimation [5]. Since our work sequentially estimates the coordinates at each time stamp, therefore we propose to predict the location displacement corresponding to a current frame for each pedestrian. Mathematically, our work aims at learning a nonlinear function $F$ by minimizing the following objective function:

$$F^* = \arg\min_F \sum_{t=obs}^{obs+T-1} \|S_{t+1} - (S_t + F(S_{1:t}))\|^2 \quad (1)$$

Here $obs$ is the number of observed frames and $F$ function as an estimation of location displacement for each pedestrian. Such location displacement or the movement from current frame to next frame is related to the pedestrian's history motion, other pedestrians' spatial affinity to the target pedestrian as well as their history motion. To model these factors for trajectory prediction, we introduce a Crowd Interaction Deep Neural Network (CIDNN) for displacement prediction. The architecture of CIDNN is depicted in Fig. 2. Specifically, CIDNN consists of four modules including motion encoder module, location encoder module, crowd interaction module, and displacement prediction module. Next, we will detail these four modules sequentially.

## 3.2. Motion Encoder Module

Motion encoder module is designed to model motion pattern of pedestrians, including different history path and direction, different velocity and acceleration. Long Short-Term Memory (LSTM) networks have been proved successful in motion modeling [1][7]. By following these work, we also employ LSTM networks to encode the motion information for each pedestrian. In our implementation, we stack two LSTM together for motion encoding. For each pedestrian, we sequentially feed the history coordinates into the stacked LSTM. For pedestrian $p_i$, we denote the output of staked LSTM as $z_t^i$ at time $t$, then mathematically

$$z_t^i = f(S_1^i, \ldots, S_t^i) \quad (2)$$

where the function $f(\cdot)$ represents the input-output function of stacked LSTM. In our implementation, the number of nodes in hidden layer of both LSTM is fixed to be 100, and all pedestrians share the same stacked LSTM for motion encoding.

## 3.3. Location Encoder Module

As aforementioned, the movement of a target pedestrian from the current frame to next frame is related to all pedestrians' motion information, including the pedestrian himself/herself as well their spatial affinity to the target pedestrian. So a straightforward way is to linearly combine the motion features of all pedestrians for displacement prediction, and the weight is based on the spatial affinity of each pedestrian to the target pedestrian, and the spatial affinity

measures the level of influence of each pedestrian to the target pedestrian.

We denote the spatial affinity between $p_i$ and $p_j$ at time $t$ as $a_t^{i,j}$, then we can use some kernel function $\kappa(S_t^i, S_t^j)$ for $a_t^{i,j}$ measurement, for example, Gaussian kernel $\kappa(S_t^i, S_t^j) = \exp(-\lambda \| S_t^i - S_t^j \|^2)$. However, such Gaussian kernel only considers the spatial distance between two pedestrians for spatial affinity measurement. It is worth noting that given two pedestrians, even their distance to the target person are the same, their spatial affinity to the target pedestrian may be different. There are two possible reasons for this: i) As shown in Fig. 1, there stands some pedestrians between $p_1$ and $p_t$, though the Euclidean distance of $p_3$ to the target is similar as that of $p_1$ to the target, but $p_1$ probably influences the trajectory of target more than $p_3$. ii) Because of the view angle of the camera, even though the distances of two pedestrian pairs calculated based on coordinates in the image are the same, it is possible the actual ground distance are different, consequently the spatial affinity of these two pedestrian pair should be different too. For example, the distance between two pedestrians in upper left corner may be the same with that of two pedestrians in lower right corner in Fig. 3, though their coordinates based distance are the same. Therefore, coordinates based spatial affinity is more meaningful than distance based affinity measurement for trajectory prediction. So is there any way to automatically learn an optical spatial affinity measurement?

The kernel trick says that $\kappa(S_t^i, S_t^j) = \left\langle \phi(S_t^i), \phi(S_t^j) \right\rangle$, here $\phi(\cdot)$ is some nonlinear function that maps the input to a high dimensional feature space, and $\langle \cdot, \cdot \rangle$ is the inner product operation. However, such $\phi(\cdot)$ is usually unknown. Motivated by the kernel trick, we propose to map the input (coordinates) into a high dimensional feature space with some neural network and use the inner product between the hidden nodes for spatial affinity measurement. Specifically, we use a multi-layer perceptron as location encoder, which contains 3 layers, and ReLU activation function is used. The number of hidden nodes in these layers is 32, 64, 128, respectively. We denote the output of location encoder for pedestrian $p_i$ at time $t$ as $h_t^i$, then

$$h_t^i = g(S_t^i) \tag{3}$$

Here $g(\cdot)$ represents the input-output function of the multi-layer perceptron of local encoder.

### 3.4. Crowd Interaction Module

Based on the output of location encoder, we can measure the spatial affinity between two pedestrians. For the a pedestrian $p_j$, we denote its spatial affinity to the target pedestrian $p_i$ at time $t$ as $a_t^{i,j}$, then

$$a_t^{i,j} = \frac{\exp(\left\langle h_t^i, h_t^j \right\rangle)}{\sum_j \exp(\left\langle h_t^i, h_t^j \right\rangle)} \tag{4}$$

It is worth noting that since $\left\langle h_t^i, h_t^j \right\rangle$ does not necessarily between [0,1], we use a softmax way to normalize it to [0,1], and use it as the affinity measurement. We can see that $a_t^{i,j}$ and $a_t^{j,i}$ are different, and this is reasonable because the movement of each pedestrian is based on himself/hereself as well as its neighbors. Even though $p_i$ is the nearest neighbour for $p_j$, but $p_j$ may not be the nearest neighbour for $p_i$. Therefore, the level of influence of $p_i$ to $p_j$ and the level of influence of of $p_j$ to $p_i$ are different.

Based on the definition of spatial affinity, we can model the level of influence of all pedestrians to person $p_i$, which is denoted as $c_t^i$, as follows:

$$c_t^i = \sum_j a_t^{i,j} z_t^j \tag{5}$$

Then we can use $c_t^i$ to predict the location displacement between time $t$ and $t+1$ for person $p_i$. Here we consider both the spatial affinity and the motion information of different pedestrian for the trajectory prediction of the target pedestrian. If the spatial affinity is larger or the pedestrian moves fast, then it is likely that the pedestrian may influence the target more.

### 3.5. Displacement Prediction Module

We use one fully connected layer with linearity to map the total effect of all pedestrians to the target $p_i$ to estimate the location displacement ($\delta S_{t+1}^i$) between time $t$ and $t+1$:

$$\delta S_{t+1}^i = W c_t^i + b \tag{6}$$

Here $W, b$ is the parameters in this fully connected layer. Once we get the location displacement, we can compute the coordinate of person $p_i$ at time $t+1$: $S_{t+1}^i = \hat{S_t^i} + \delta S_{t+1}^i$.

It is also worth noting that we predict the trajectory for each pedestrian separately. Therefore our framework can be easily paralleled in implementation. Further, compared with Behavior CNN [25] and Social LSTM [1], the number of hidden nodes in our framework is very small. Therefore our method is very efficient in implementation, especially when the number of pedestrians is small in a scene.

## 4. Experiments

### 4.1. Experimental Setup

We implement our solutions with the PyTorch framework, and mini-batch based stochastic gradient descent is

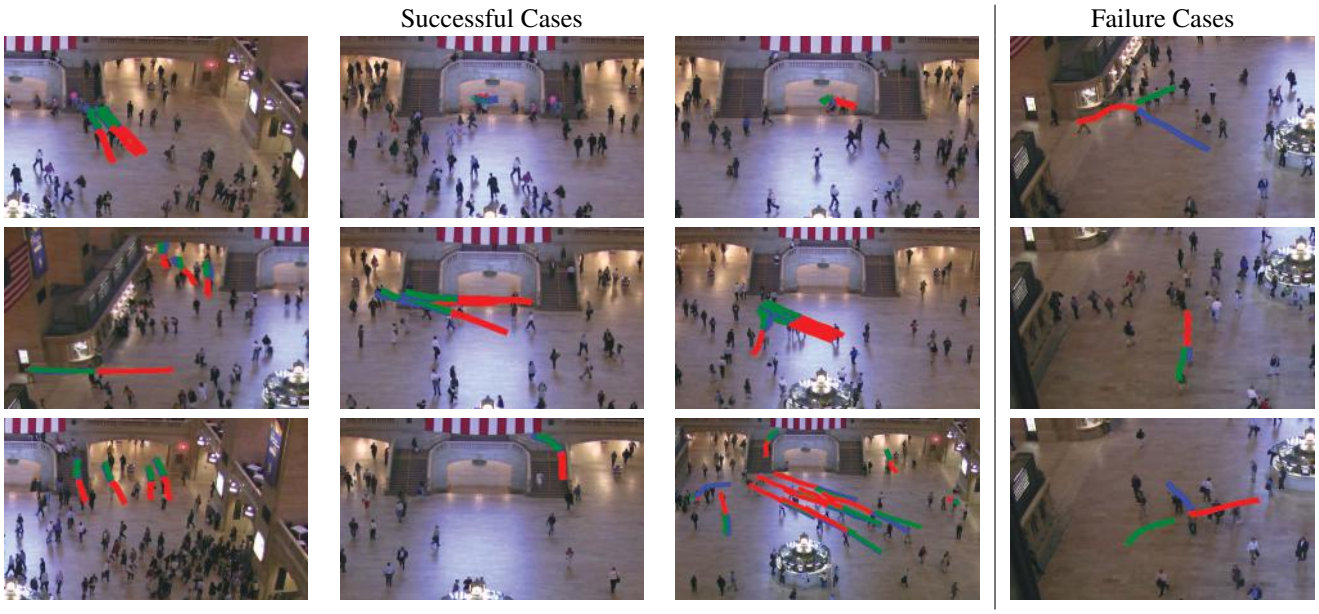



Figure 3. Qualitative results: history trajectory (red), ground truth (blue), and predicted trajectories from our model (green). The first three columns show some successful cases and last column shows some failure cases. We can see that our prediction always overlaps with ground truth, which shows the effectiveness of our method. Please enlarge the figure for better visualization.

used to optimize the objective function. We train our network with the following hyper-parameters setting: mini-batch size (256), learning rate (0.003), momentum (0.9), weight decay (0.005), and number of epochs (50,000). The parameters are initialized with 'Xavier'.

**Datasets.** We evaluate our method with the following publicly available human trajectory datasets: New York Grand Central (GC) [24], ETH [18], UCY [15], the CUHK Crowd Dataset [20] and the subway station dataset[30]. As shown in [18], these datasets also cover very challenging group behaviors such as couples walking together, groups crossing each other and groups forming and dispersing in some scenes.

**The GC dataset** consists of around 12,600 pedestrians and it is about one hour long. By following the same experimental setup with [25], 4990 short clips are uniformly segmented from GC dataset, and one sample can be obtained from each clip. The first 90% samples are used for training while the remaining for test.

**The ETH dataset** contains two scenes each with 750 different pedestrians split into two sets (ETH and Hotel).

**The UCY dataset** includes two scenes with 786 people. This dataset has 3-components: ZARA-01, ZARA- 02 and UCY. These datasets represent crowded real-world settings with thousands of non-linear trajectories.

**The CUHK Crowd Dataset** contains many crowd videos with different densities and perspective scales in many environments.

**The subway station dataset** is a 30-minute sequence collected in the New York Grand Central Station, with each containing more than 40,000 keypoint trajectories in total. Following the same experimental setup and evaluation criteria as [1], we use a leave-one-out approach on the 5 sets of ETH and UCY. We train and validate our model on 4 sets and test on the remaining set. We repeat this for all the 5 sets. We also use the same training and testing procedure for other baseline methods used for performance comparison.

**Measurement.** By following the work [25], we use Average Displacement Error (ADE) as metric to measure the performance of different methods. ADE is the mean square error (MSE) overall estimated points of a trajectory and the true points. It can be mathematically defined as follows:

$$ADE = \frac{\sum_i \sum_{t=obs}^{obs+T-1} \|S_{t+1}^i - \hat{S}_{t+1}^i\|_2}{nT} \quad (7)$$

In our experiments, we have observed the trajectory for 5 frames and use them to predict the trajectory for the next 5 frames, therefore $obs = 5$ and $T = 5$. Actually the data on the GC dataset is sampled from real videos with time interval 20 frames, so the time interval between two neighbouring frames is 0.8 sec, and the prediction of the 5th frame is the coordinates in the coming 4 sec.

**Baselines.** Following the experimental setup in [25], we design the following baselines:

i) The constant acceleration regressors were used to predict future walking path of each pedestrian, and this baseline is termed as const acc; We also compare our method with the

| dataset | const acc | SF [23] | S-LSTM [1] | B-CNN [25] | SRGP [7] | Ours |
|---|---|---|---|---|---|---|
| ETH | 0.80 | 0.41 | 0.50 | 0.35 | NA | 0.09 |
| HOTEL | 0.39 | 0.25 | 0.11 | 0.18 | NA | 0.11 |
| ZARA 1 | 0.47 | 0.40 | 0.22 | 0.20 | NA | 0.15 |
| ZARA 2 | 0.45 | 0.40 | 0.25 | 0.23 | NA | 0.10 |
| UCY | 0.57 | 0.48 | 0.27 | 0.25 | NA | 0.12 |
| GC | 0.099 | 0.033 | 0.020 | 0.024 | NA | 0.012 |
| CUHK Crowd | 0.046 | NA | 0.0341 | NA | 0.029 | 0.008 |
| subway station | 0.064 | NA | 0.0335 | NA | 0.031 | 0.016 |

Table 1. The performance comparison of different methods on the GC, ETH, UCY, CUHK Crowd and subway station datasets.

following state-of-the-art baselines *:
ii) Social force (SF) [23] which employs an agent for each pedestrian to simulate the trajectory making process. iii) Behaviour-CNN (***B-CNN***) [25] where a deep neural network (Behavior-CNN) is proposed to model pedestrian behaviors in crowd scenes;
iv) Social LSTM (***S-LSTM***) [1] where generated multiple LSTMs for each pedestrian are used to estimate their positions considering the neighboring pedestrians.
v) SRGP [7] where long short-term memory (LSTM) networks with social-aware recurrent Gaussian processes is used to model the complex transitions and uncertainties of the crowd.

## 4.2. Performance Comparison

We compare our method and other baseline methods on GC, ETH, UCY, the CUHK Crowd Dataset and the subway station dataset in Table 1. We can see that our method significantly outperforms all existing methods and other baselines on all the datasets, which validates the effectiveness of our solution. The comparison based metrics used in [1] *Final displacement error (FDE)* and *Average non-linear displacement error (ANDE)* is listed in Table 2.

| Methods | ADE | | FDE | | ANDE | |
|---|---|---|---|---|---|---|
| | GC | Subway | GC | Subway | GC | Subway |
| B-CNN | 0.024 | NA | 0.0495 | NA | 0.0284 | NA |
| S-LSTM | 0.020 | 0.0335 | 0.0456 | 0.045 | 0.0403 | 0.0403 |
| Ours | **0.012** | **0.008** | **0.0229** | **0.0266** | **0.0257** | **0.0299** |

Table 2. Comparisons on more evaluation metric.

We further show the predicted trajectory and its ground truth on the GC dataset in Fig. 3. As aforementioned, every pedestrians trajectory will be influenced by near other people, but our crowd interaction module can learn different patterns of this influences. The first three columns show that the model can well predict the trajectories even if they intersect with others. Meanwhile, in the last column of Fig. 3, we also show some failure cases, which is probably due to the sudden changes of destination in ground truth. Even though such sudden change is hard to model, our method still predicts very similar trajectory compared with ground truth.

*Because the same experimental setup of [1], [25] and same training/testing sets are used, we directly adapted the results of baselines from [1], [25]. The performance of Social LSTM [1], Behavior CNN [25], SRGP [7], and SF [23] on these datasets are directly adapted from the corresponding papers.

Besides spatial affinity, the movement of someone also depends on his movement velocity, direction, etc. We show two examples in Fig. 4. We can see that the one walks slow would let the one walks fast go first for both ground-truth and prediction in collision cases.


Figure 4. Examples of collision case.

## 4.3. Evaluation of Different Components in CIDNN

**Coordinate regression vs. displacement regression** In our displacement prediction module, we use a fully connected layer to map the weighted features to estimate the displacement $\delta S^i_{t+1}$ between time $t+1$ and time $t$ for pedestrian $p_i$. Besides the $\delta S$ regression, we also try to use the weighted features to directly estimate $S^i_{t+1}$. We show results of these two different strategies in Table 3. We can see that displacement regression always achieves higher accuracy than directly predicting the ground truth. This is because the LSTM encodes the velocity, acceleration between continuous frames well, and it is easier to predict the displacement only than predict (displacement + current coordinates). The good performance of displacement regression strategy validates the effectiveness of residual regression in trajectory prediction, which agrees with existing work for image classification and facial/body key points detection.

**With Crowd Interaction module vs. Without Crowd Interaction module** To validate the effectiveness of crowd interaction module, we also train a network without the crowd interaction module, i.e., we directly estimate the displacement based on motion features extracted from stacked LSTM. We compare the performance with/without crowd interaction module in Table 3. We can see that that network with crowd interaction module performs better than the one without crowd interaction module. This is because the network with crowd interaction module takes the different importance of different neighboring pedestrians into consideration.

**The evaluation of motion encoder** To evaluate the importance of motion encoder, we propose to replace it with the displacements between all previous neighboring frames in CIDNN, and we term such baseline as CIDNN w/o LSTM. The comparison between our method and CIDNN w/o LSTM is listed in Table 3. We can see that our CIDNN achieves better performance, which validates the effectiveness of LSTM for motion characterization.

| Datasets | Regression Strategy | | Crowd Interaction | |
|---|---|---|---|---|
| | Coordinate | Displacement | With | Without |
| ETH | 0.23 | 0.09 | 0.09 | 0.11 |
| ZARA 2 | 0.24 | 0.10 | 0.10 | 0.15 |
| GC | 0.018 | 0.012 | 0.012 | 0.055 |
| Datasets | Spatial Affinity | | Motion Encoder | |
| | Ours | Gaussian | With LSTM | Without LSTM |
| ETH | 0.09 | 0.11 | 0.09 | 1.37 |
| ZARA | 0.10 | 0.16 | 0.10 | 0.17 |
| GC | 0.012 | 0.055 | 0.012 | 0.018 |

Table 3. Performance evaluation of different components in CIDNN.

We also investigate the performance of stacked LSTM with different layers for motion encoder on GC. When the number layer of stacked LSTM is 1, 2, and 3, the MSE results are 0.014, 0.0125, and 0.013.Considering accuracy and efficiency, we fix the layers to be 2.

**Different spatial affinity measurement** To evaluate the importance of our location encoder, we also compare our method with Gaussian kernel $\kappa(S_t^i, S_t^j) = \exp(-\lambda \| S_t^i - S_t^j \|^2)$, which is a distance based spatial affinity measurement. The performance comparison is shown in Table 3. The good performance of our method validates the effectiveness of our location encoder module for spatial affinity measurement.

We further show the relationship between our spatial affinity and the Euclidean distance between two pedestrian on GC in Fig. 5 (a). We can see that the spatial affinity measured by our method is usually larger for points with smaller distance, and smaller for points with larger distance. Sa shown in Fig. 5 (b), since the Euclidean distance between pedestrian pair ($P_2$ $P_{T2}$), and ($P_3$, $P_{T2}$) is similar, and the spatial affinity between them is also similar, while the Euclidean distance between pairs ($P_2$ $P_{T2}$) is less than that of ($P_4$ $P_{T2}$), the spatial affinity of ($P_4$ $P_{T2}$) is smaller. Further, as we discussed earlier, even two pedestrian pairs are with different distances computed based coordinates, their spatial affinity may be similar because coordinates based distance is not the actual ground plane distance due to effect of view angle. We also give an example in Fig. 5 (b), and we can see that even though the coordinate based distance between pedestrian pair ($P_1$, $P_{T1}$) is smaller than that of pedestrian pair ($P_2$, $P_{T2}$), but because of view angle, their ground distance is similar, so their spatial affinity is also similar. Fig. 6 shows spatial affinity for different pedestrians at different time stamps in a scene on the GC dataset. We can see that neighboring pedestrians are usually with larger spatia affinity, which validates the effectiveness of our spatial affinity definition.

**The performance on challenging data** To show the performance of different methods for more challenging data, we split the GC dataset into two subsets (non-straight/straight trajectories). Results are shown in 4. Results show our method achieves the best performance on both subsets.

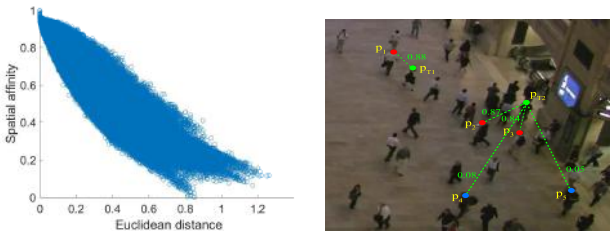



Figure 5. (a) The relationship between the our spatial affinity and Euclidean distance on the GC dataset. We normalize the coordinates to [0,1] when we feed the coordinates to location encoder. So the distance is also normalized. (b) An instance for illustrating the relationship between Euclidean distance and spatial affinity.

| Method | straight subset (97.8%) | non-straight subset (2.2%) |
|---|---|---|
| Ours | **0.0123** | **0.0206** |
| B-CNN | 0.0284 | 0.0361 |
| S-LSTM | 0.0540 | 0.0254 |

Table 4. Prediction errors for straight non-straight trajectories.

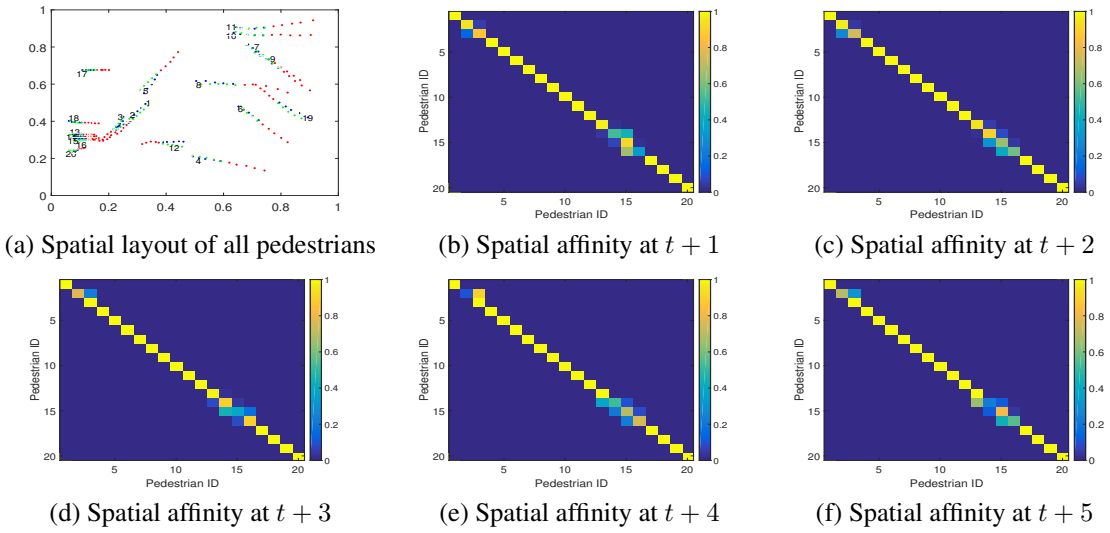
We randomly contaminate a fraction of training data with gaussian noise $\mathcal{N}(0, \frac{\bar{v}}{3})$, where $\bar{v}$ is their mean velocity. When gaussian noise is 0%, 5%, 10%, and 20%, the MSE results on GC dataset is 0.0125, 0.0137, 0.0143, and 0.0145, which shows the robustness of our method.

**Trajectory prediction for a longer time** The ADE of trajectory prediction for 0.8 sec, 4 sec, and 8 sec is 0.005, 0.012, and 0.034, respectively, on GC, which is better than that of social LSTM, which achieves 0.009, 0.020, and 0.040, respectively. We further see that the performance of trajectory prediction degenerates for a longer time. But the improvement of our method over social LSTM increases as time goes longer because our formulation considers all possible pedestrians which may contribute the targets trajectory prediction in future.

**The input of motion encoder** In our implementation, we feed the coordinates of each pedestrian at different time into the motion encoder. We also try to replace the input of motion encoder with the displacement between neighboring frames on GC datasets. Such model achieves a score of 0.021 in terms of ADE on GC, while our coordinates based model achieves 0.012 in terms of ADE. One possible reason is that the coordinates of pedestrians would provide extra location information apart from motion, which further boosts the performance of trajectory prediction.

### 4.4. Transferability of location encoder and motion encoder

To evaluate the transferability of location encoder and motion encoder, we conduct cross domain experiments. We




(a) Spatial layout of all pedestrians (b) Spatial affinity at $t+1$ (c) Spatial affinity at $t+2$

(d) Spatial affinity at $t+3$ (e) Spatial affinity at $t+4$ (f) Spatial affinity at $t+5$

Figure 6. Crowd Interaction scores for different motion encoders at different time. (a) represents qualitative results: history trajectory (red), ground truth (blue), and predicted trajectories with our model (green). (b) - (f) represent spatial affinity scores at time from $t+1$ to $t+5$ respectively. $x$ axis and $y$ axis represent the ID of each pedestrian.

| Setting | | Transferred components | | | Ours |
|---|---|---|---|---|---|
| Source | Target | M and L | M | L | |
| subway | GC | 0.085 | 0.021 | 0.034 | 0.012 |
| GC | subway | 0.068 | 0.018 | 0.023 | 0.016 |
| GC | ETH | 1.19 | 0.03 | 0.1 | 0.09 |
| GC | HOTEL | 0.83 | 0.057 | 0.099 | 0.11 |
| GC | ZARA1 | 1.28 | 0.034 | 0.045 | 0.15 |
| GC | ZARA2 | 1.54 | 0.035 | 0.046 | 0.1 |
| GC | UCY | 0.90 | 0.073 | 0.114 | 0.12 |

Table 5. Prediction errors under different transfer learning settings.

evaluate the performance of location encoder ($L$) and motion encoder ($M$) trained on target domain with the target domain. In our experiments, we use GC and subway station because of they both corresponds to subway scenes, and choose one dataset as source domain and use the other as target domain, as shown in Table. 5. Since GC is much lager than subway station, and the model trained on GC also achieves satisfactory results on subway. Further, we found motion is easier to transfer because all possible motion trend of all pedestrians can be well covered by a larger dataset. By contrast, the transferability of location encoder is not so good because the scene layout as well as the camera perspective are different. Further, the performance of the model trained on GC and then finetuned on subway is 0.017, and 0.013 if source/target is changed reversely.

### 4.5. Time cost

We test the running time of our method on the GC dataset. Our model is implemented on an NVIDIA GeForce TITAN GPU platform and an Intel(R) Xeon(R) CPU E5-2643 v3 3.40GHz CPU platform, respectively. We run our program 20 times and calculate the average running time for each image. More precisely, the average running time of CIDNN is 0.43 ms on GPU. The time cost of CIDNN is 1.91 ms on CPU.

## 5. Conclusion

In this paper, we design a crowd interaction deep neural network (CIDNN) for displacement prediction. Our model considers the difference level of influence of different pedestrians in the crowd on the target pedestrian. Specifically, we propose to use LSTM to model the motion of each pedestrian, then we weight the motion feature of all pedestrians based on their spatial affinity to the target pedestrian for location displacement prediction. Compared with existing work Social LSTM[1], Behaviour-CNN[25], our method considers the different importance of all pedestrians based on their spatial affinity to the target pedestrian. Extensive experiments on publicly available datasets validate the effectiveness of our method for trajectory prediction.

The proposed solution here is used for trajectory prediction. But it also can be applied to other applications, for example, facial keypoint detection in videos and human pose estimation (body keypoint detection) in videos. Appearance based key point detection is usually time-consuming. By combining our method with appearance based keypoint detection in key frames, we can avoid keypoint detection for each frame, which may improve the efficiency without reducing accuracy.

## 6. Acknowledgement

This project is supported by the NSFC (No. 61502304) and Program of Shanghai Subject Chief Scientist (A type) (No.15XD1502900).

## References


[1] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 961–971, 2016.

[2] G. Antonini, M. Bierlaire, and M. Weber. Discrete choice models of pedestrian walking behavior. *Transportation Research Part B*, 40(8):667–687, 2006.

[3] E. Bonabeau. Agent-based modeling: Methods and techniques for simulating human systems. *Proceedings of the National Academy of Sciences*, 99(suppl 3):7280–7287, 2002.

[4] B. Cancela, A. Iglesias, M. Ortega, and M. G. Penedo. Unsupervised trajectory modelling using temporal information via minimal paths. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2553–2560, 2014.

[5] J. Carreira, P. Agrawal, K. Fragkiadaki, and J. Malik. Human pose estimation with iterative error feedback. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4733–4742, 2016.

[6] R. Emonet, J. Varadarajan, and J.-M. Odobez. Extracting and locating temporal motifs in video scenes using a hierarchical non parametric bayesian model. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3233–3240. IEEE, 2011.

[7] Y. D. B. Z. Hang Su, Jun Zhu. Forecast the plausible paths in crowd scenes. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*, pages 2772–2778, 2017.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[9] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European Conference on Computer Vision*, pages 630–645. Springer, 2016.

[10] D. Helbing and P. Molnar. Social force model for pedestrian dynamics. *Physical review E*, 51(5):4282, 1995.

[11] T. M. Hospedales, J. Li, S. Gong, and T. Xiang. Identifying rare and subtle behaviors: A weakly supervised joint topic model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(12):2451–2464, 2011.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[13] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: a partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD international conference on Management of data*, pages 593–604. ACM, 2007.

[14] 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. pages 2165–2174, 2017.

[15] A. Lerner, Y. Chrysanthou, and D. Lischinski. Crowds by example. In *Computer Graphics Forum*, volume 26, pages 655–664. Wiley Online Library, 2007.

[16] R. Mehran, A. Oyama, and M. Shah. Abnormal crowd behavior detection using social force model. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 935–942. IEEE, 2009.

[17] B. Morris and M. Trivedi. Learning trajectory patterns by clustering: Experimental studies and comparative evaluation. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 312–319. IEEE, 2009.

[18] S. Pellegrini, A. Ess, K. Schindler, and L. Van Gool. You'll never walk alone: Modeling social behavior for multi-target tracking. In *Computer Vision, 2009 IEEE 12th International Conference on*, pages 261–268. IEEE, 2009.

[19] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[20] J. Shao, C. Change Loy, and X. Wang. Scene-independent group profiling in crowd. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2219–2226, 2014.

[21] X. Wang, K. T. Ma, G.-W. Ng, and W. E. L. Grimson. Trajectory analysis and semantic region modeling using nonparametric hierarchical bayesian models. *International journal of computer vision*, 95(3):287–312, 2011.

[22] X. Wang, X. Ma, and W. E. L. Grimson. Unsupervised activity perception in crowded and complicated scenes using hierarchical bayesian models. *IEEE Transactions on pattern analysis and machine intelligence*, 31(3):539–555, 2009.

[23] K. Yamaguchi, A. C. Berg, L. E. Ortiz, and T. L. Berg. Who are you with and where are you going? In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1345–1352. IEEE, 2011.

[24] S. Yi, H. Li, and X. Wang. Understanding pedestrian behaviors from stationary crowd groups. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3488–3496, 2015.

[25] S. Yi, H. Li, and X. Wang. Pedestrian behavior understanding and prediction with deep neural networks. In *European Conference on Computer Vision*, pages 263–279. Springer, 2016.

[26] S. Yi, X. Wang, C. Lu, J. Jia, and H. Li. $l\_0$ regularized stationary-time estimation for crowd analysis. *IEEE transactions on pattern analysis and machine intelligence*, 39(5):981–994, 2017.

[27] C. Zhang, H. Li, X. Wang, and X. Yang. Cross-scene crowd counting via deep convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 833–841, 2015.

[28] J. Zhang, S. Shan, M. Kan, and X. Chen. Coarse-to-fine auto-encoder networks (cfan) for real-time face alignment. In *ECCV*, pages 1–16. Springer, 2014.

[29] B. Zhou, X. Tang, and X. Wang. Learning collective crowd behaviors with dynamic pedestrian-agents. *International Journal of Computer Vision*, 111(1):50–68, 2015.

[30] B. Zhou, X. Wang, and X. Tang. Random field topic model for semantic region analysis in crowded scenes from tracklets. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 3441–3448. IEEE, 2011.

[31] B. Zhou, X. Wang, and X. Tang. Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 2871–2878. IEEE, 2012.