IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 27, 2020, accepted February 20, 2020, date of publication March 2, 2020, date of current version March 13, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2977747

A Location-Velocity-Temporal Attention LSTM Model for Pedestrian Trajectory Prediction

HAO XUE, DU Q. HUYNH, (Senior Member, IEEE), AND MARK REYNOLDS, (Member, IEEE)
Department of Computer Science and Software Engineering, The University of Western Australia, Perth, WA 6009, Australia

Corresponding author: Hao Xue (hao.xue@research.uwa.edu.au)

The work of Hao Xue was supported by the International Postgraduate Research Scholarship (IPRS) at The University of Western Australia.

ABSTRACT Pedestrian trajectory prediction is fundamental to a wide range of scientific research work and industrial applications. Most of the current advanced trajectory prediction methods incorporate context information such as pedestrian neighbourhood, labelled static obstacles, and the background scene into the trajectory prediction process. In contrast to these methods which require rich contexts, the method in our paper focuses on predicting a pedestrian's future trajectory using his/her observed part of the trajectory only. Our method, which we refer to as LVTA, is a Location-Velocity-Temporal Attention LSTM model where two temporal attention mechanisms are applied to the hidden state vectors from the location and velocity LSTM layers. In addition, a location-velocity attention layer embedded inside a *tweak* module is used to improve the predicted location and velocity coordinates before they are passed to the next time step. Extensive experiments conducted on three large benchmark datasets and comparison with eleven existing trajectory prediction methods demonstrate that LVTA achieves competitive prediction performance. Specifically, LVTA attains 9.19 pixels Average Displacement Error (ADE) and 17.28 pixels Final Displacement Error (FDE) for the Central Station dataset, and 0.46 metres ADE and 0.92 metres FDE for the ETH&UCY datasets. Furthermore, evaluation on using LVTA to generate trajectories of different prediction lengths and on new scenes without the need of retraining confirms that it has good generalizability.

INDEX TERMS Pedestrian trajectory prediction, long short-term memory (LSTM), attention models, human movement analysis.

I. INTRODUCTION

Trajectory prediction is essential for a wide range of applications such as forecasting trajectories of vulnerable road users in traffic environments [1] and location based services [2], [3]. It is also an important component for Advanced Driver Assistance Systems (ADAS) and autonomous vehicles [4].

One way to predict pedestrians' trajectories in a scene is to model the physics of the human movement patterns. A classical paper is the social force model (SFM) [5] which uses two different types of forces to capture these patterns: the attractive forces which pull people towards their destinations; and the repulsive forces which keep people apart and away from obstacles in the scene. In the last few years, there has been a surge of interest in pedestrian trajectory prediction and various new methods have been reported. Methods that were proposed prior to 2015 follow the SFM method by exploring the physical aspects of crowd movement; for instance, by minimizing collisions among pedestrians [6], by modelling the pedestrian dynamics in terms of interaction forces and potential energies of particles [7], by taking into account small obstacles, like vending machines, dumpster, etc, in the scene that influence the pedestrian trajectories [8], or by modelling human motion using ensemble Kalman filtering [9]. With the booming of data-driven deep learning networks, we see a vast growth recently in the literature of pedestrian trajectory prediction, focusing on the use of Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM) networks [10]–[21]. However, the concept of modelling human-human interaction and human-scene relationship remains in these new methods and has been implemented in the form of social pooling layers or social pooling modules of the network [11], [22], [23]. Some of these new methods incorporate additional information like the head poses of the pedestrians [17] or give a more explicit treatment to the scene contexts such as static obstacles [24], labelled entrance/exit regions [25], and even the whole background scenes [12], [18].

While the methods mentioned above produce promising prediction results, they require the neighbouring pedestrians to be captured alongside each person of interest (POI) both

The associate editor coordinating the review of this manuscript and approving it for publication was Xian Sun.

spatially and temporally. Not only does this requirement increase the computation time and storage space, a proper neighbourhood size around the POI must be defined for each scene in order for these methods to produce their best performance. This further reduces the *generalizability* of these methods, i.e., the methods cannot be directly applied to scenes on which the methods were not trained. Obviously, if scene information is not available, methods that require scene information as input cannot be applied either.

Contrary to the methods outlined above, simpler yet similarly effective methods have also been studied. One example is the trajectory prediction method of Nikhil and Morris [14]. To achieve real-time performance, their method uses parallelizable convolutional layers and incorporates no social or scene information. Another example is the method of Schöller *et al.* [26] where the authors revisit and use the simple constant velocity model to predict the relative displacements between consecutive location points. In our previous work [27], apart from the velocity information computed directly from the input trajectories, our joint ***L**ocation **V**elocity **A**ttention* LSTM based network (LVA) also requires no neighbourhood or scene information. For ADAS and driverless vehicle applications where computation resources are limited, these simpler methods are more preferrable.

In this paper, we extend our previous LVA method by incorporating two attention mechanisms, which we refer to as *temporal attention*, to appropriately weight the hidden state vectors output by the location-LSTM and velocity-LSTM layers of the network. We name our proposed trajectory prediction method *LVTA*, where '*T*' stands for the added temporal attention mechanism. Our LVTA method has the advantage that the prediction process only depends on the trajectory of the POI. Neither scene information nor neighbouring trajectories is required in LVTA. Instead, a tweak module is used to fuse the location and velocity information captured in the observed part of the POI's trajectory. Because of that, the architectures of both methods do not have pooling layers or pooling modules as used in [11], [22], [23]. As shown in our previous work [27], LVA already has good prediction performance, our extensive experiments confirm that the inclusion of the temporal attention mechanisms significantly improves the performance of LVTA and its generalizability. Specifically, our proposed LVTA outperforms LVA when both methods are trained and tested on the same scene (dataset). In addition, LVTA has better generalizability, as demonstrated from its superior performance on forecasting predictions on a new, unseen scene. For different prediction lengths of trajectories, LVTA also consistently outperforms LVA. Furthermore, compared to several recent trajectory prediction methods, LVTA achieves state-of-the-art prediction performance on two large benchmark datasets.

In summary, our research contributions are:

- Our proposed architecture has two LSTM layers to capture the embeddings of location and velocity coordinates of trajectories. It does not rely on scene information and has good generalizability.
- Our architecture has a module, which includes a location-velocity attention layer, to tweak the outputs from the LSTM layers. As demonstrated in our ablation study, the tweak module helps to give significant improvement to the prediction results.
- The temporal attention mechanism incorporated in our LVTA method is inspired by the work in machine translation. It captures the relationship of the hidden state vectors between the observed and predicted parts of trajectories. Our experiments show that temporal attention helps further improve the prediction performance.

The rest of the paper is organized as follows. Section II gives an overview of the related work on trajectory prediction. Section III details our LVTA architecture and the two main attention mechanisms. Section IV begins with an outline of the datasets and the metrics used in the experiments. Detailed implementation, including hyperparameter tuning, ablation study, generalizability study, and comparison with state-of-the-art methods take up a large part of this section. Also included in the section is the computation times of LVA and LVTA. Finally, the paper is concluded in Section V.

II. RELATED WORK

In this section, we give a brief review of the literature on pedestrian trajectory prediction, focusing especially on techniques that we compare with our proposed method. A large number of existing methods employ context information such as human-human interaction (*e.g.*, [11], [16], [20], [22], [28]) and/or human-space interaction (*e.g.*, [12], [18], [21], [24], [29], [30]). Similar to our work, there are also methods that incorporate attention mechanisms. We therefore group existing methods into two broad categories, namely methods with context and methods with attention, in the two subsections below.

A. TRAJECTORY PREDICTION WITH CONTEXT

The Social LSTM model of Alahi *et al.* [11] is one of the early trajectory prediction methods based on the sequence generation model from Graves [31]. Their model combines the behaviour of other people within a local neighbourhood of the POI. Another piece of early work is the behaviour-CNN method of Yi *et al.* [32], where the pedestrians' walking paths are encoded and predicted in the form of 3D displacement volumes using a number of convolutional layers. As each displacement volume contains all pedestrians at the same time period, the behaviour-CNN is able to capture their influence to each other. Following the work of Alahi *et al.* [11], and with the success of the Generative Adversarial Network (GAN) [33] in other applications [34], [35], Gupta *et al.* [22] propose the Social GAN model (abbreviated as SGAN) to generate multiple trajectory predictions for each input observed trajectory. Their network includes a pooling module to expand the neighbourhood around each POI to cover the whole scene so all the pedestrians can be considered

in the training and prediction processes. This effectively expands the local neighbourhood context to a global level. The Social-Aware Generative Adversarial Imitation Learning (SA-GAIL) method of Zou *et al.* [25] is another example based on GAN. The authors combined a collision avoidance regularization and the Social LSTM into the trajectory prediction process. Also using the LSTM architecture, the SR-LSTM method [20] handles pedestrians' interaction as a message passing process among them.

Incorporating scene context as well in the trajectory prediction process often requires the coding of the scene features as part of the network architecture; for example, apart from encoding the neighbourhood information, Xue *et al.* [18] use a deep CNN to encode scene context in their hierarchical LSTM network; Liang *et al.* [21] use a pretrained scene segmentation model to extract a number of semantic scene classes and compute their scene CNN features using two convolutional layers in their network. Similar to Liang *et al.*'s work, SoPhie [30] also extracts semantic scene features but it uses the raw features from the VGGnet-19 network and projects them to a lower dimension. Like the SGAN model [22], SoPhie uses GAN to generate multiple future paths for each input trajectory.

B. TRAJECTORY PREDICTION INCLUDING ATTENTION

The effectiveness of attention mechanism was first demonstrated by Bahdanau *et al.* [36] in the neural machine translation task. Luong *et al.* [37] later improved the machine translation performance by designing different score functions (*e.g.*, the *dot*, *general*, and *concat* score functions) for calculating the attention scores. Since then, attention based deep learning models have been widely used in other tasks, such as image captioning [38], video captioning [39], action recognition [40], [41], person re-identification [42], [43], and time series data classification [44], [45].

In the area of trajectory prediction, attention mechanisms have been used for capturing the relative importances of neighbours around a person [46] and for learning the embedding information of a person's own trajectory plus the context information of surrounding neighbours [47]. Different from the above two papers, Sadeghian *et al.* [48] focus on the problem of a wider scope: one that involves both pedestrians and vehicles. The authors use the term *agent* to denote a human or a vehicle in a scene. An attention mechanism was used in their network for input scene images to highlight the important regions for each agent's future path. For the SoPhie method [30] mentioned above, two attention modules are used to deal with scene context and social interactions.

Our LVTA differs from the methods reviewed above in that it does not require neighbouring and scene information when the method predicts the future trajectory of each POI. It uses both temporal attention and location-velocity mechanisms on the trajectory embeddings. This is different from the attention mechanism used in [30], [46], [47].

III. METHODOLOGY

A. PROBLEM FORMULATION

We represent the trajectory of the i^{th} pedestrian as a time sequence of two dimensional coordinates (x_t^i, y_t^i), obtained via a tracking method or manual labelling. The coordinates can be in metres on the ground or in image pixel unit. The aim of trajectory prediction is to use the observed locations of the i^{th} pedestrian, for all i, from time $t = 1$ to $t = T_{\text{obs}}$ to predict locations from $t = T_{\text{obs}} + 1$ to $t = T_{\text{obs}} + T_{\text{pred}}$, where T_{obs} and T_{pred} denote, respectively, the lengths of the observed and predicted trajectories. We represent these two segments of each trajectory as $\mathbf{X}_{\text{obs}}^i = \left[\left(x_1^i, y_1^i\right), \cdots, \left(x_{T_{\text{obs}}}^i, y_{T_{\text{obs}}}^i\right)\right]$ and $\mathbf{X}_{\text{pred}}^i = \left[\left(x_{T_{\text{obs}}+1}^i, y_{T_{\text{obs}}+1}^i\right), \cdots, \left(x_{T_{\text{obs}}+T_{\text{pred}}}^i, y_{T_{\text{obs}}+T_{\text{pred}}}^i\right)\right]$.

From the observed trajectory $\mathbf{X}_{\text{obs}}^i$, the velocity information $\mathbf{U}_{\text{obs}}^i = \left[\left(u_1^i, v_1^i\right), \cdots, \left(u_{T_{\text{obs}}}^i, v_{T_{\text{obs}}}^i\right)\right]$ is obtained from the finite differences of $\mathbf{X}_{\text{obs}}^i$ over the time steps. We duplicate the first velocity term $\left(u_1^i, v_1^i\right)$ so that $\mathbf{U}_{\text{obs}}^i$ has the same number of time steps as $\mathbf{X}_{\text{obs}}^i$. The velocity information computed above is equivalent to the *relative coordinates* used in some implementation [49] in the literature. In this paper, our interest is to generate the predicted trajectory $\mathbf{X}_{\text{pred}}^i$ based on the combined location and velocity information $\left\{\mathbf{X}_{\text{obs}}^i, \mathbf{U}_{\text{obs}}^i\right\}$ of the observed trajectory. To simplify the explanation in the later subsections, the superscript i is dropped from hereon.

B. THE ARCHITECTURE OF LVTA

As shown in Figure 1, two separate LSTM layers are used in the proposed LVTA architecture: a location LSTM layer and a velocity LSTM layer. These two layers, each of which is of hidden dimension N_{h}, process the embedding vectors $\mathbf{e}_t^l$ and $\mathbf{e}_t^v$ of the location and velocity coordinates of the observed trajectories in parallel. To simplify the visualization, the embedding layers of embedding size N_{e} are omitted in the figure. Thus, starting with the location and velocity coordinates (x_t, y_t) and (u_t, v_t) at time t, we have a series of equations listed below:

$$\mathbf{e}_t^l = \phi^l\left(x_t, y_t; \mathbf{W}_e^l\right), \tag{1}$$

$$\mathbf{h}_{t+1}^l = \text{LSTM}^l\left(\mathbf{h}_t^l, \mathbf{e}_t^l; \mathbf{W}^l\right), \tag{2}$$

$$\left(\hat{x}_t, \hat{y}_t\right)^\top = \mathbf{W}_o^l \mathbf{h}_t^l + \mathbf{b}_o^l, \tag{3}$$

$$\mathbf{e}_t^v = \phi^v\left(u_t, v_t; \mathbf{W}_e^v\right), \tag{4}$$

$$\mathbf{h}_{t+1}^v = \text{LSTM}^v\left(\mathbf{h}_t^v, \mathbf{e}_t^v; \mathbf{W}^v\right), \tag{5}$$

$$\left(\hat{u}_t, \hat{v}_t\right)^\top = \mathbf{W}_o^v \mathbf{h}_t^v + \mathbf{b}_o^v. \tag{6}$$

In Eqs. (1)-(6), $\text{LSTM}^l(\cdot)$ and $\text{LSTM}^v(\cdot)$ represent the location LSTM layer and velocity LSTM layer (the white and blue boxes in Figure 1). Functions $\phi^l(\cdot)$ and $\phi^v(\cdot)$ are embedding functions for the location and the velocity coordinates. The $\mathbf{W}$ terms are the different weight matrices, *e.g.*, $\mathbf{W}_e^l$ and $\mathbf{W}_e^v$ denote the weight matrices of two embedding layers; $\mathbf{W}^l$ and $\mathbf{W}^v$ denote the weight matrices of the two LSTM layers.

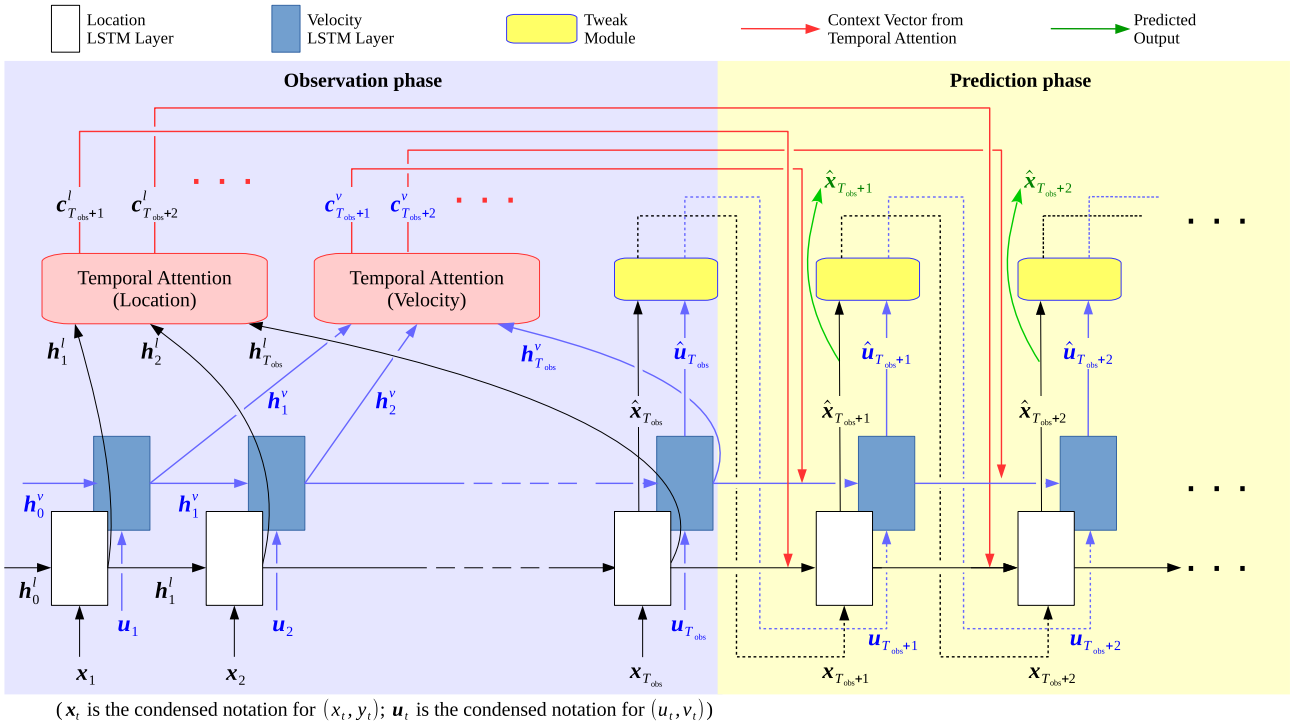


FIGURE 1. Our proposed LVTA network. Two LSTM layers are used for the location and the velocity embeddings separately. For each LSTM layer, a temporal attention mechanism is used to generate context vectors. In the prediction phase, the outputs from the location and velocity LSTM layers are modified by a *tweak module* before they are passed to the next time step. To simplify the visualization, the embedding vectors $\mathbf{e}_t^l$ and $\mathbf{e}_t^v$, for all t, are not shown.

Similarly, all the **b** terms denote the bias vectors and the **h** terms denote the hidden states of LSTM layers.

The embedding layer and the two LSTM layers in the network are designed to capture the latent representation of trajectories and the complex pedestrian movement patterns in the scene. The network also includes two attention mechanisms (red boxes in Figure 1) for the location and velocity LSTM layers. The job of these attention mechanisms is to combine the latent information captured in the observation phase of each trajectory to yield better predicted trajectories in the prediction phase.

In the observation phase ($t = 1$ to T_{obs}), (x_t, y_t) and (u_t, v_t) are directly acquired from each observed trajectory. In the prediction phase ($t = T_{\text{obs}} + 1$ to $T_{\text{obs}} + T_{\text{pred}}$), the inputs at time step t come from the *tweak module* which operates on the inputs and predicted outputs at time step $t - 1$. The subsections below detail the attention mechanisms of the LVTA architecture.

C. TEMPORAL ATTENTION

The temporal attention mechanism captures the relationships between a time step of the prediction phase and different time steps of the observed part of the input trajectory. In a nutshell, it outputs a different weight for each of these relationships based on the input that is passed to it. As a result, this extra information in temporal attention is able to help produce more accurate prediction and improve the robustness in predicting trajectories of different lengths.

To simplify the description, the remaining part of this subsection focuses only on the temporal attention mechanism for the location LSTM layer. The mechanism for the velocity LSTM layer is similar.

Following the use of context vectors in machine translation [36] and in trajectory prediction [47] to capture different attentions at different time steps of the input sequence, we introduce the context vector $\mathbf{c}_t^l$ to the LVTA architecture. For the trajectory observation phase, the hidden states $\mathbf{h}^l = \left[\mathbf{h}_1^l, \ldots, \mathbf{h}_{T_{\text{obs}}}^l\right]$ are computed from Eq. (2). During the trajectory prediction phase ($T_{\text{obs}} + 1 \leqslant t \leqslant T_{\text{pred}}$), Eq. (2) becomes:

$$\mathbf{h}_{t+1}^l = \text{LSTM}^l\left(\mathbf{h}_t^l, \mathbf{c}_t^l, \mathbf{e}_t^l; \mathbf{W}^l\right), \tag{7}$$

where the context vector $\mathbf{c}_t^l$ is defined in terms of all the hidden states in the observation phase, *i.e.*,

$$\mathbf{c}_t^l = \sum_{s=1}^{T_{\text{obs}}} \beta_{s,t}^l \mathbf{h}_s^l. \tag{8}$$

All the $\beta_{s,t}^l$ terms in the equation above are the attention weights between time step $s \leqslant T_{\text{obs}}$ and time step $t > T_{\text{obs}}$ that need to be computed to yield a good estimate of each $\mathbf{c}_t^l$. They are defined in terms of the hidden state vectors. In the paper of Luong *et al.* [37], the authors describe three different score functions to model the relationship between $\mathbf{h}_s^l$ and $\mathbf{h}_t^l$. To get the value of $\beta_{s,t}^l$, we adopt their *general score* function

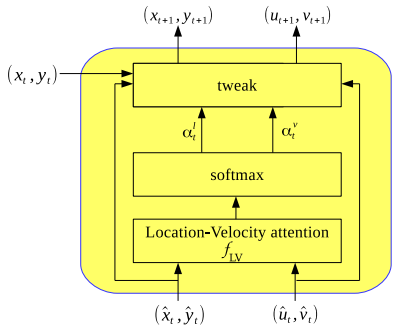


FIGURE 2. Details of the tweak module shown in Figure 1. It has three layers: a location-velocity attention layer, a softmax layer, and a tweak layer.

given by f_{T} below:

$$f_{\mathrm{T}}\left(\mathbf{h}_s^l, \mathbf{h}_t^l\right) = \mathbf{h}_s^{l\top} \mathbf{W}_{\mathrm{T}}^l \mathbf{h}_t^l \tag{9}$$

as we found it suitable in capturing the correlation between different components of $\mathbf{h}_t$. Here, $\mathbf{W}_{\mathrm{T}}^l \in \mathbb{R}^{N_{\mathrm{h}} \times N_{\mathrm{h}}}$ is a weight matrix that needs to be trained. The softmax function is then used to normalized all the $\beta_{s,t}^l$ terms so that $\sum_{s=1}^{T_{\mathrm{obs}}} \beta_{s,t} = 1$, *i.e.*,

$$\beta_{s,t}^l = \frac{\exp f_{\mathrm{T}}\left(\mathbf{h}_s^l, \mathbf{h}_{t-1}^l\right)}{\sum_{k=1}^{T_{\mathrm{obs}}} \exp f_{\mathrm{T}}\left(\mathbf{h}_{t-1}^l, \mathbf{h}_k^l\right)}. \tag{10}$$

The LVTA architecture thus has two extra weight matrices, $\mathbf{W}_{\mathrm{T}}^l$ and $\mathbf{W}_{\mathrm{T}}^v$, that require training in the training phase for the location and velocity temporal attention mechanisms.

D. THE TWEAK MODULE

Our LVTA architecture differs from the conventional LSTM sequence generation models such as the widely used *Seq2Seq* model [50] and the *Encoder-Decoder* model [51]. In conventional LSTM models, the output of the LSTM network at the last time step is directly used as input for the next time step in the decoder phase. In our proposed LVTA architecture, each time step produces two outputs: location coordinates and velocity coordinates. We can therefore use a *tweak module* (yellow boxes in Figures 1 and 2) to refine them at time step t before passing them on to time step $t+1$.

Let $(\hat{x}_t, \hat{y}_t)$ and $(\hat{u}_t, \hat{v}_t)$ be the output location and velocity coordinates predicted by the LSTM^l and LSTM^v layers at time step t. Rather than using them as inputs for time $t+1$, they are passed to the tweak module. The role of the tweak module is to feed $(\hat{x}_t, \hat{y}_t)$ and $(\hat{u}_t, \hat{v}_t)$ through a few layers to yield better location and velocity coordinates (x_{t+1}, y_{t+1}) and (u_{t+1}, v_{t+1}) for the next time step. The tweak module consists of three layers (Figure 2):

- The location-velocity (LV) attention layer, denoted by f_{LV}, is implemented as a linear fully connected layer with 4 input neurons and 2 output neurons. If a more complex model is desired, the attention layer can be easily replaced by, for instance, a multilayer perceptron.
- The output from the LV attention layer is then passed through a softmax activation layer to yield two weight parameters, α_t^l and α_t^v as follows:

$$\left(\alpha_t^l, \alpha_t^v\right) = \mathrm{softmax}\left(f_{\mathrm{LV}}\left(\hat{x}_t, \hat{y}_t, \hat{u}_t, \hat{v}_t\right)\right). \tag{11}$$

  The softmax function is used so that α_t^l and α_t^v can be treated as probability values, *i.e.*, $0 \leqslant \alpha_t^l, \alpha_t^v \leqslant 1$, and $\alpha_t^l + \alpha_t^v = 1$.
- With α_t^l and α_t^v from Eq. (11) above and input location coordinates (x_t, y_t), the job of the final *tweak* layer is to compute the input location and velocity coordinates for time step $t+1$ using the following equations:

$$x_{t+1} = \alpha_t^l \hat{x}_t + \alpha_t^v \left(x_t + \hat{u}_t\right), \tag{12}$$
$$y_{t+1} = \alpha_t^l \hat{y}_t + \alpha_t^v \left(y_t + \hat{v}_t\right), \tag{13}$$
$$u_{t+1} = x_{t+1} - x_t, \tag{14}$$
$$v_{t+1} = y_{t+1} - y_t. \tag{15}$$

The location-velocity attention mechanism implemented in the tweak module described above allows the model to learn α_t^l and α_t^v through the model training process, update (x_t, y_t) and (u_t, v_t) at each time step, and keep track of the relationship between the location and velocity information along the way. Its realisation introduces only a small 4×2 weight matrix that requires training.

The results from Eqs. (12)-(15) are fed into the network at time step $t+1$. The procedure described above is repeated until the time step $t = T_{\mathrm{pred}}$ is reached.

It should be noted that the two types of attention mechanisms described so far differ in two respects:

1) The two temporal attention mechanisms (see the previous subsection) operate on the location hidden state vectors and velocity hidden state vectors separately, whereas the location-velocity attention layer inside the tweak module (Figure 2) operates on the location and velocity coordinates together.
2) The temporal attention mechanisms capture the relationship of the hidden state vectors between the observation phase and the prediction phase, whereas the location-velocity attention layer captures the relationship between the location and velocity coordinates at the prediction phase only.

IV. EXPERIMENTS

A. DATASETS AND METRICS

1) CENTRAL STATION DATASET

This large publicly available dataset [52] contains over 10,000 trajectories extracted from a 33 minutes long surveillance video. The scene resolution is 720 (width) $\times$ 480 (height) in pixels. We preprocessed the dataset by normalizing all the trajectory coordinates so that they are in the [0, 1] range. We use 10-fold cross-validation to evaluate the performance of our method. Each fold is in turn used as the test set while the remaining folds are used as the

training set. The average performance from the 10 folds is reported for this dataset.

2) ETH/UCY DATASET

The combined ETH [6] and UCY [53] dataset is another widely used public dataset for evaluating trajectory prediction methods. It contains a total of 5 scenes, which have over 1,000 trajectories altogether, known as ETH, HOTEL (both are from ETH), UNIV, ZARA1, and ZARA2 (the last three are from UCY). We followed the common leave-one-out evaluation policy that has been used in [11], [20], [22], [30], *i.e.*, we trained on four scenes and tested on the remaining scene. The labelled coordinates of each pedestrian are given in metres. The prediction results of our method and its variants reported later in the paper are the performance on the test set.

3) EDINBURGH DATASET

This dataset consists of trajectories of people walking around the Informatics Forum at the University of Edinburgh [54]. It covers several months of observations, resulting in over 92,000 trajectories in total. Same as [47], 20,000 trajectories and 5,000 trajectories were randomly sampled to form the training set and the test set. The scene images are 640×480 pixels, where each pixel covers a 24.7mm $\times$ 24.7mm region on the ground. This dataset was used specifically for evaluating the generalizability of the proposed architecture, *i.e.*, we trained LVTA using the Central Station dataset and tested its prediction performance on the test set of this dataset.

4) EVALUATION METRICS

Similar to the previous work [10], [11], [17], [18], [22], [25], [27], [29], we use the average displacement error (ADE) [6] and the final displacement error (FDE) [11] metrics to quantitatively evaluate the trajectory prediction performance of each method. The former is the mean Euclidean distance between all the points in the predicted and ground truth trajectories averaged over all the trajectories. The latter is the average Euclidean distance between their final points (or the destination points). Where appropriate, we also use the *normADE*, which is the ADE being normalized with respect to the image size.

B. IMPLEMENTATION DETAILS

Our LVTA method and its variants were trained by the Adam optimizer [55] with 0.001 learning rate for 500 epochs with a mini batch size $m = 128$. The loss function $\mathcal{L}$ is given by:

$$\mathcal{L}(\mathbf{X}_{\text{gt}}, \mathbf{X}_{\text{pred}}) = \frac{1}{m} \sum_{i=1}^{m} \|\mathbf{X}^i_{\text{gt}} - \mathbf{X}^i_{\text{pred}}\|^2, \tag{16}$$

where $\mathbf{X}^i_{\text{gt}}$ and $\mathbf{X}^i_{\text{pred}}$ denote the ground truth trajectory and predicted trajectory of the i^{th} pedestrian in the mini batch. As the velocity term for each time step is not used for the ADE and FDE computation, it is not included in the loss function.

Our proposed LVTA and its variants were implemented[1] using the Pytorch framework in Python and trained with an NVIDIA GeForce GTX-1080 GPU. For some of the experiments on the ETH/UCY dataset, we used an NVIDIA Titan XP GPU.

1) HYPERPARAMETER TUNING

We focus on three hyperparameters that are crucial to the performance of the proposed trajectory prediction method: the dropout rate, the hidden dimension N_{h} of the LSTM layers, and the embedding dimension N_{e} of the embedding layers.

To investigate how these three hyperparameters influence the performance of LVTA, all the combination of the following hyperparameter values were evaluated:

- N_{h}: [32, 64, 128, 256]
- N_{e}: [64, 128, 256]
- dropout: [0.1, 0.2, 0.5]

resulting in 36 experiments in total. The above sets of values for the hidden dimension and embedding dimension are chosen based on the values used in other trajectory prediction methods in the literature, *e.g.*, $\{N_{\text{h}} = 128, N_{\text{e}} = 64\}$ is used in [11] and $\{N_{\text{h}} = 256, N_{\text{e}} = 128\}$ is used in [21].

For this hyperparameter tuning process, we randomly selected from the Central Station dataset 80% and 10% of the trajectories to form the training and validation sets. The ADE and FDE for different combinations of hyperparameter values on the validation set are shown in Figure 3. From left to right, the dropout rate is 0.1, 0.2 and 0.5 for each column. In each subfigure, a lighter background colour means a lower error (*i.e.*, better prediction performance). It can be seen in Figure 3 that the prediction performance of LVTA becomes worse when the hidden size N_{h} (same for the embedding size N_{e}) is either too small or too large. These results are expected as N_{h} and N_{e} determine the number of parameters (weights of each layer in the network) that require training. Large N_{h} and N_{e} values result in more parameters and may lead to overfitting problems, whereas small N_{h} and N_{e} values do not allow the algorithm to model the complex trajectory patterns. For all the experiments reported in the rest of this section, both the embedding size N_{e} and the hidden size N_{h} were therefore set to 128 and the dropout rate was set to 0.5, as this hyperparameter value combination (see Figure 3) has the lowest ADE (9.17) and FDE (17.04) on the validation set.

Figure 4 illustrates the loss plot of the training set and validation set for this optimal hyperparameter setting. The figure shows how the mean squared error (MSE) varies as the number of epochs increases. In the region of 300-500 epochs, the training curve (blue) continues to drop but the decrease in MSE is very small whereas the validation curve (red) fluctuates slightly and comes to a plateau slowly. The two curves are very close to each other, indicating that the network is well behaved and that there is no overfitting issue. The MSE that

[1] Codes can be found: https://github.com/xuehaouwa/LVTA.

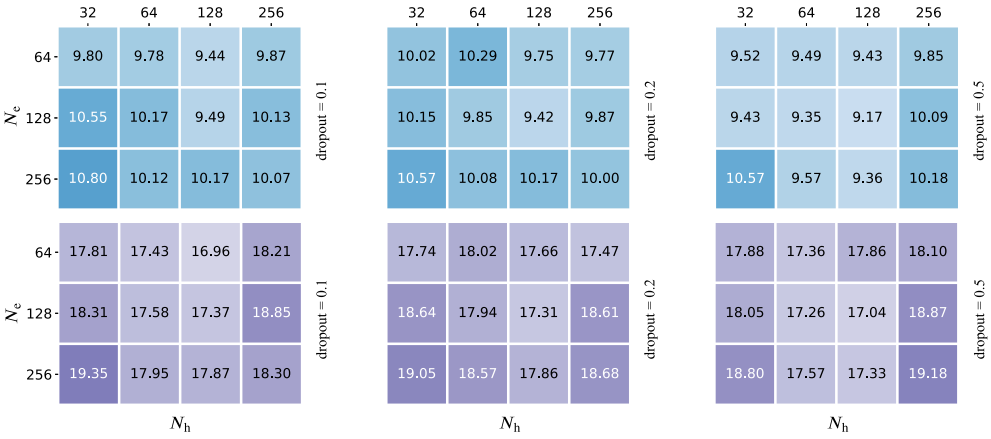


FIGURE 3. ADE (top row) and FDE (bottom row) on the validation set of the Central Station dataset under different hyperparameter combinations. From left to right, the dropout rate is 0.1, 0.2, and 0.5 for each column.

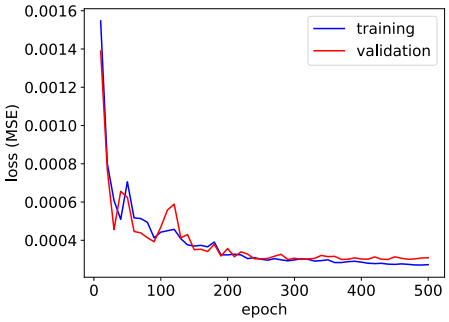


FIGURE 4. A loss plot showing the decrease of Mean Squared Error (MSE) over the number of training epochs for the training and validation sets. The optimal hyperparameter values used in the plot are: $N_e = N_h = 128$; dropout rate = 0.5.

is minimized in the loss function (Eq. 16) is analogous to the average displacement error (ADE) in trajectory prediction.

2) ABLATION STUDY

We evaluate the prediction performance of three variants of our LVTA method:

a: THE VANILLA LV MODEL

This variant is the LVTA architecture with both the tweak module and the temporal attention mechanism removed. It can be considered as a modified vanilla LSTM model. The difference between the traditional vanilla LSTM model and the *vanilla LV* model is the input and output of the network. For the vanilla LSTM, the input and output at each time step are 2D location vectors of the trajectory in the form (x_t, y_t); for the vanilla LV model, these are vectors of the form (x_t, y_t, u_t, v_t), i.e., $\mathbf{X}_{\text{pred}} \in \mathbb{R}^{T_{\text{pred}} \times 4}$ and the predicted trajectory is obtained by extracting the (x_t, y_t) coordinates from the 4D vectors.

b: THE CONSTANT LOCATION-VELOCITY ATTENTION MODEL (CLVA)

This variant of LVTA has the tweak module (Figure 2) modified by removing both the location-velocity attention layer (f_{LV}) and the softmax layer. Rather than training the network to learn f_{LV} for the optimal values for α_t^l and α_t^v, the CLVA model has both α_t^l and α_t^v set to 0.5 for all the time steps in the prediction phase, i.e., $\alpha_t^l = \alpha_t^v = 0.5$, for $T_{\text{obs}} + 1 \leqslant t \leqslant T_{\text{obs}} + T_{\text{pred}}$.

c: THE TEMPORAL ATTENTION MODEL (LVT)

This variant is the LVTA architecture with only the tweak module removed. Thus, the location-velocity attention mechanism (see Figure 2) inside the tweak modeule is not included either. LVT contains the two temporal attention mechanisms shown in Figure 1.

d: THE LOCATION-VELOCITY ATTENTION MODEL (LVA)

This variant does not have the two temporal attention mechanisms. It is the same method originally proposed in our previous work [27]. LVA contains the full tweak module, *i.e.*, including the location-velocity attention mechanism.

3) METHODS BEING COMPARED

We compare the performance of our LVTA method as well as its four variants (*vanilla LV, CLVA, LVT, and LVA*) against 12 methods listed below: *Linear*, *Social Force Model (SFM)* [5], *Linear Trajectory Avoidance (LTA)* [6], *Behaviour CNN* [32], *Vanilla LSTM*, *SA-GAIL* [25], *Social-LSTM* [11], *Attention-LSTM* [47], *SGAN* [22], *Nikhil and Morris* [14], *Liang et al.* [21], and *SR-LSTM* [20].

Depending on the dataset and availability of results, not all methods were compared on all datasets.

C. RESULTS ON THE CENTRAL STATION DATASET

Table 1 shows the trajectory prediction results on the test set of the Central Station dataset. We followed the setting used in [25] and fixed $T_{\text{obs}} = 9$ and $T_{\text{pred}} = 8$. The results show that the constant velocity method is only able to learn relatively straight path trajectories and performs very poorly. With the attractive forces (moving towards destinations) and the repulsive forces (avoiding collision with other people or obstacles) incorporated into the models, both the Social Force

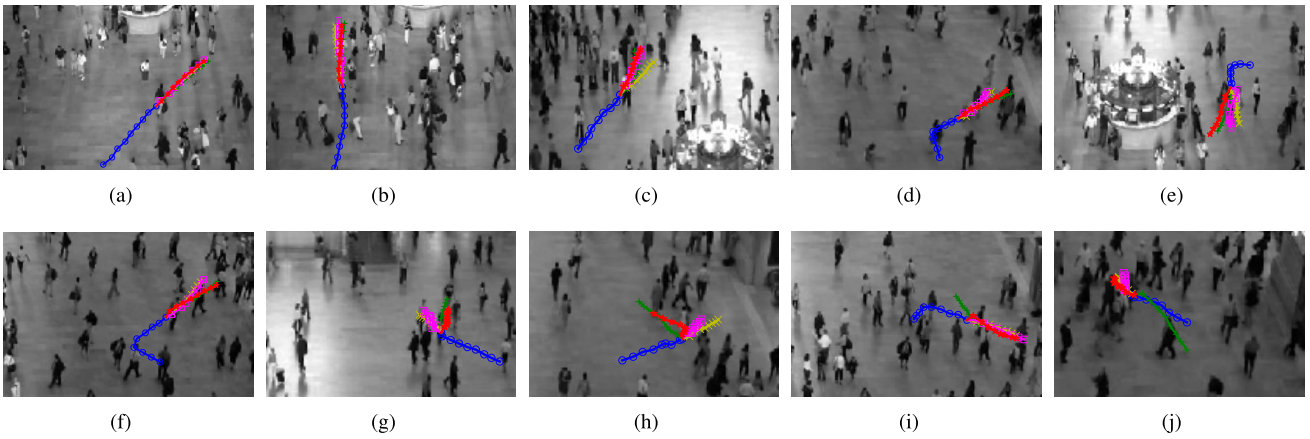


FIGURE 5. (a) - (j) Qualitative comparison of prediction results from vanilla LSTM, LVA, and LVTA. Colour codes: blue: input observed trajectories; green: ground truth trajectories; yellow: prediction by vanilla LSTM; pink: prediction by LVA; red: prediction by LVTA. The background scene used in each subfigure is the first frame of the prediction phase. Two failure cases are given in the last two subfigures ((i) & (j)).

TABLE 1. Prediction errors of different prediction methods on the Central Station dataset.

method	normADE	ADE	FDE
Linear*	5.86%	–	–
SFM* [5]	4.45%	–	–
LTA* [6]	4.35%	–	–
Vanilla LSTM*	2.39%	14.57	27.78
Behaviour CNN* [32]	2.52%	–	–
SA-GAIL* [25]	1.98%	11.98	23.05
Our vanilla LV	2.23%	13.78	23.97
Our CLVA	2.09%	12.29	21.74
Our LVT	1.88%	10.96	20.56
Our LVA	1.65%	10.05	19.43
Our LVTA	**1.55%**	**9.38**	**17.22**

* results extracted from [25]

method and the LTA method give more superior performance than the constant velocity method. It is evident from Table 1 that the more recent deep learning based prediction methods (the last 8 rows of the table) are way ahead of the former 3 prediction methods by a large margin. Comparing with the Behaviour CNN method, even the vanilla LSTM gives a smaller prediction error. The results on this dataset show that, for time sequence data, LSTM based methods are more suitable than CNN based methods.

The prediction results of vanilla LV, CLVA, LVT, LVA, and LVTA are given in the last five rows of Table 1. Our LVTA outperforms other baselines and the state-of-the-art methods on all the three metrics. LVA takes the second place after LVTA. Although the prediction result of CLVA is worse than SA-GAIL, LVT, LVA, and LVTA, it performs better than other baseline methods. The extra velocity information being passed to vanilla LV demonstrates to be useful as the prediction result from vanilla LV is slightly better than that from vanilla LSTM. Compared to CLVA (having 2.09% normalized ADE), the location-velocity attention layer in the tweak module helps LVTA gain significant improvement (at 1.55% normalized ADE) on the prediction results. If the tweak module is completely removed from the network architecture as in the variant LVT, the performance is worst than LVTA. This further confirms that the two temporal attention mechanisms and the tweak module in the LVTA architecture work well together and help to reduce the prediction errors.

Figure 5 shows a qualitative comparison of some prediction results from the vanilla LSTM, LVA, and LVTA models. For simple cases (Figure 5(a) and (b)) where the trajectories are almost linear, all three methods generate trajectories very close to the ground truth trajectories. With the inclusion of the velocity LSTM layer, both LVA and LVTA are able to give more precise predicted trajectories in the examples of turning slightly (Figure 5(c)) and turning abruptly (Figures 5(d)-(f)) captured in the observation phase. Although LVA and vanilla LSTM both give reasonably good trajectory directions, LVTA generates more accurate predicted trajectories that almost overlap with the ground truth trajectories.

Extremely challenging cases are shown in Figure 5(g) and (h). Unlike the turning cases in Figures 5(d)-(f), the turning occurs very late in the observed part (blue colour) of each trajectory. Although not completely coinciding with the ground truth trajectories, LVTA still manages to predict the turning trend and generate plausible trajectories than the vanilla LSTM and LVA methods. These challenging examples demonstrate the effectiveness in the prediction phase of the temporal attention mechanism that is present in LVTA but absent in LVA.

Two failure cases are shown in Figure 5(i) and (j). There are two reasons for the failed predictions in these two examples. First, the turning movements occur after the observation part so, using the observed trajectories alone, all the three methods fail to predict these late changes of walking direction. Second, both ground truth trajectories in the prediction phase are very sharp U-turns (almost 180°) that are very different from the gentle turns shown in Figures 5(d)-(f) earlier where LVTA is

TABLE 2. Prediction errors (in metres) of different prediction methods on the ETH/UCY dataset. The results marked with * are taken from [22]. Top-1, top-2, top-3 results are shown in blue, red, and green. The results from SR-LSTM [20] come from the best 20m × 20m neighbourhood region reported by the authors.

Datasets	Performance (ADE / FDE in metres)							
	Linear*	Vanilla LSTM*	Social-LSTM* [11]	SGAN [22] 1V-1*	Nikhil & Morris [14]	SR-LSTM [20]	Liang *et al.* [21]	LVTA (ours)
ETH	1.33 / 2.94	1.09 / 2.41	1.09 / 2.35	1.13 / 2.21	1.04 / 2.07	0.63 / 1.25	0.88 / 1.98	0.57 / 1.10
HOTEL	0.39 / 0.72	0.86 / 1.91	0.79 / 1.76	1.01 / 2.18	0.59 / 1.17	0.37 / 0.74	0.36 / 0.74	0.42 / 0.69
UNIV	0.82 / 1.59	0.61 / 1.31	0.67 / 1.40	0.60 / 1.28	0.57 / 1.21	0.51 / 1.10	0.62 / 1.32	0.55 / 1.19
ZARA1	0.62 / 1.21	0.41 / 0.88	0.47 / 1.00	0.42 / 0.91	0.43 / 0.90	0.41 / 0.90	0.42 / 0.90	0.42 / 0.92
ZARA2	0.77 / 1.48	0.52 / 1.11	0.56 / 1.17	0.52 / 1.11	0.34 / 0.75	0.32 / 0.70	0.34 / 0.75	0.35 / 0.75
Average	0.79 / 1.59	0.70 / 1.52	0.72 / 1.54	0.74 / 1.54	0.59 / 1.22	0.45 / 0.94	0.52 / 1.14	0.46 / 0.92

still able to give plausible predictions. To improve the prediction results for these cases, more training data having late and sudden changes of walking direction would be required. It may also help if higher order terms such as acceleration are incorporated into the network architecture.

D. RESULTS ON THE ETH/UCY DATASET

For each test scene of this dataset, we first combined the trajectories of all the training scenes. We then carried out a normalization step by setting the origin at the centroid of the trajectories and scaled them so that all the trajectory coordinates are in the range $[-1, 1]$. The same normalization parameters were applied to the test set. After performing trajectory prediction on the test set, the inverse normalization was applied to yield the ADE and FDE in metres.

We followed the setting used in [11], [20], [22], [30] and fixed $T_{\text{obs}} = 8$ and $T_{\text{pred}} = 12$. Compared to the other two datasets, the number of trajectories in this dataset is relatively small. As in [20], we therefore augmented the training set by

- splitting long trajectories to form trajectories of length $T_{\text{obs}} + T_{\text{pred}} = 20$ using a sliding time window of stride size 1; and
- performing random rotation.

In addition, we also performed trajectory reversal and swapped the x and y-coordinates (equivalent to 90° rotation). Trajectory rotation is in general not used for data augmentation as the augmented trajectories might not be valid (*e.g.*, the trajectories might represent pedestrians walking into obstacles). However, since this dataset is about using training trajectories captured from up to four different scenes to predict trajectories in the fifth scene, trajectories are not bound to any scene context. It should be noted also that, for our LVTA method and its variants, trajectory reversal does not provide extra training information to the velocity LSTM layer as the velocity coordinates of a reversed trajectory are simply the negated version of the original trajectory; however, reversed trajectories do provide extra information to the training of the location LSTM layers of these models.

Table 2 shows the prediction performance of our LVTA versus two baseline methods (*Linear* and *Vanilla LSTM*) and five state-of-the-art methods on the five scenes. On each row, the top three performing methods are highlighted in blue, red, and green. The last row of the table shows the average performance over the five scenes. Although both the SGAN method [22] and Liang *et al.*'s method [21] can generate multiple trajectory predictions through their GAN based architectures and have better prediction results, for fair comparison, we only include their single prediction performances in the table. On the other hand, SoPhie [30] is not included in the comparison in Table 2 because its single prediction results are not reported by the authors. For the SR-LSTM method [20], multiple results with different configurations have been reported in [20] and the authors' best prediction results are shown in the table.

Unlike the traditional CNN architecture which focuses on the spatial information only, the CNN based prediction method of Nikhil and Morris [14] uses parallelizable convolutional layers to handle temporal dependencies. Their method appears to give prediction results that are comparable with other LSTM based methods.

For the ETH scene, our LVTA method outperforms all other methods with a 0.57m ADE and 1.10m FDE, leading a comfortable margin from the runner-up SR-LSTM. For the other four scenes, our LVTA's ADEs and FDEs are among the top three methods. On average, LVTA takes the first spot on FDE at 0.92m and the second spot on ADE at 0.46m, which is only 0.01m behind the winner SR-LSTM.

It should be noted that the training and test trajectories of all the five scenes in the ETH/UCY dataset cover roughly a 24m × 24m region. The SR-LSTM model incorporates a 20m × 20m neighbourhood region and a pedestrian-wise attention layer to model the influence from other pedestrians. This large neighbourhood region used in SR-LSTM is almost the entire scene. This means that much computational work is needed in SR-LSTM to store the hidden states of other pedestrians. Compared to SR-LSTM, LVTA is computationally more efficient as only the POI information is required to predict its trajectory. Furthermore, when the neighbourhood region shrinks to 4m × 4m, the average prediction errors of SR-LSTM increase to 0.49m for ADE and 1.06m for FDE (These results are reported in Table 2 of [20]). If we use these errors in our Table 2 instead, our average ADE will move to the first place, ahead of SR-LSTM.

Some prediction results from the ETH/UCY datasets are illustrated in Figure 6. The image coordinates of the overlaid

FIGURE 6. Illustration of predicted trajectories on the ETH and HOTEL scenes of the ETH/UCY dataset. Colour codes: blue: input observed trajectories; green: ground truth trajectories; pink: prediction by LVTA.

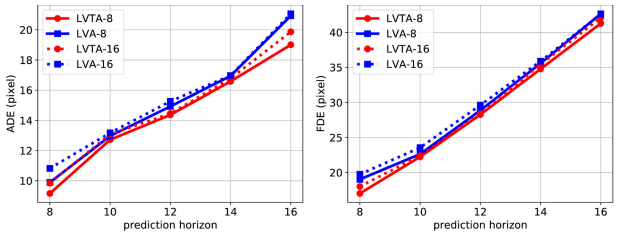


FIGURE 7. The ADE and FDE of prediction results from our LVA and LVTA methods for different prediction horizons (T_{ph}). Dash lines represent models that are trained with $T_{pred} = 16$ and solid lines are models trained with $T_{pred} = 8$. Results of using LVA and LVTA are shown in blue and red respectively.

trajectories are converted from the homography matrix provided for each scene in the dataset. The figure shows that LVTA can generate plausible trajectories for different cases such as stopping and slowing down.

E. GENERALIZABILITY STUDY

1) PREDICTING TRAJECTORIES OF DIFFERENT PREDICTION HORIZONS

To distinguish prediction lengths of trajectories in the training and testing stages, we adopt the term *prediction horizon* [23], denoted by T_{ph} from hereon, to mean the prediction length in the testing stage. The objective of the experiments in this section is to evaluate how well the LVTA architecture can be generalized to produce trajectories of different prediction horizons.

For the LVA and LVTA models that have been trained to predict n time steps, i.e., $T_{pred} = n$, we represent them as LVA-n and LVTA-n. Once a network is trained, it can be used to predict trajectories of any T_{ph} value. With two settings $\{T_{obs} = 9, T_{pred} = 8\}$ and $\{T_{obs} = 9, T_{pred} = 16\}$ on the training set of the Central Station dataset, we trained four separate models: LVA-8, LVTA-8, LVA-16, and LVTA-16. Their performance on predicting trajectories of various prediction horizon values were then compared on the test set.

Prior to the training stage, trajectories of a suitable length need to be extracted from the training set. For LVA-16, trajectories must be at least $T_{obs} + 16 = 9 + 16 = 25$ time steps long; for LVA-8, they only need to have at least $T_{obs} + 8 = 9 + 8 = 17$ time steps. In total, only 9,328 trajectories could be used to train LVA-16; however, 14,739 trajectories were available to train LVA-8. Thus, while one might expect that a network that is trained to predict trajectories of a longer prediction length should perform better than one that is trained to predict trajectories of a shorter prediction length, it is not always the case as the former network is exposed to fewer, and therefore less diverse, trajectories.

Figure 7 shows the ADEs and FDEs of the predicted trajectories from the four models mentioned above on the test set for 5 different T_{ph} values: 8, 10, 12, 14, and 16. As expected, both the ADE and FDE increase with increasing T_{ph}. Comparing LVA-8 with LVA-16, it shows that it is not always an advantage to train a network with large T_{pred} when T_{ph} is small, *e.g.*, LVA-16 performs worse than LVA-8 when $T_{ph} = 8$. Only when T_{ph} increases to 16 that LVA-16 slightly outperforms LVA-8. However, when comparing LVTA-8 against LVTA-16, we do not see the same pattern. For small T_{ph} values (*e.g.*, when $T_{ph} = 8$), LVTA-8 outperforms LVTA-16 as expected. Furthermore, LVTA-8 is able to maintain its superior performance even when $T_{ph} \geqslant 12$. This demonstrates the effectiveness of the extra temporal attention mechanism in the architecture.

Some example trajectories generated by LVTA-8 and LVTA-16 for the Central Station test set are shown as red and pink trajectories in Figure 8. The ground truth trajectories for the prediction phase and the input observed trajectories are given in green and blue. The first row has prediction horizon $T_{ph} = 8$ and the second row has $T_{ph} = 16$. It can be seen from the figure that LVTA-16 performs better than LVTA-8 in some occasional turning cases when $T_{ph} = 16$. However, at the beginning of the prediction phase, the location coordinates predicted by LVTA-8 are more accurate. On average, LVTA-8 has smaller ADE and FDE than LVTA-16 for all the T_{ph} values in our experiments (Figure 7).

2) TRANSFERRING TO OTHER SCENES

Recall that the Edinburgh dataset has more training trajectories than the Central Station dataset (see Section IV-A). So it will be more advantageous to train a prediction model on the Edinburgh dataset and test it on the same dataset. The aim of the experiments conducted in this subsection is to show the generalizability of the LVTA architecture on transferring what is learned from a scene to another scene. It should be noted that, except for ZARA1 and ZARA2, the experiments on the ETH/UCY dataset described in the previous subsection are also tests on the generalizability as the training scenes and the test scene are different. Since existing methods report their performance on the Edinburgh dataset in metres, to be consistent with them, we use the pixel to metre relationship described in Section IV-A to convert the ADE and FDE values from pixels to metres.

Without any retraining, the four models, namely LVA-8, LVA-16, LVTA-8, and LVTA-16, that have been trained on the Central Station dataset (in pixels) described in Section IV-C are directly used to predict trajectories in the test set of the Edinburgh dataset. To be consistent with the test

FIGURE 8. Prediction results of different prediction horizons: $T_{ph} = 8$ frames (first row) and $T_{ph} = 16$ frames (second row) on the Central Station test set. Colour codes: blue: input observed trajectories; green: ground truth trajectories; red: LVTA-8; pink: LVTA-16. The background scene in each subfigure is the first frame of the prediction phase.

TABLE 3. Error performance (in metres) of different prediction methods on the Edinburgh dataset.

method	transfer	ADE	FDE
SFM* [5]		3.124	3.909
Social-LSTM* [11]		1.524	2.510
Attention-LSTM* [47]		0.986	1.311
LVA-8	✓	1.194	2.176
LVA-16	✓	1.540	2.417
LVTA-8	✓	0.890	1.706
LVTA-16	✓	1.182	1.914

* results extracted from [47]

results from other methods reported in [47], T_{obs} and T_{ph} were both set to 20. In Table 3, a method having a tick under the *transfer* column denotes that it has been trained on the Central Station training set instead of the Edinburgh training set. The prediction results show that LVTA-8 and LVTA-16 have similar performance as Attention-LSTM while both LVTA-8 and LVTA-16 outperform the LVA counterparts. Even not being trained on trajectories from the same scene, LVTA-8 outperforms all the other techniques on the ADE.

To further investigate the generalizability of LVA and LVTA, we test the two pretrained models above on the Edinburgh test set with the T_{obs} value ranging from 7 to 21 and T_{ph} ranging from 6 to 18. It is clear in Figure 9 that the ADEs and FDEs of both LVTA-8 and LVTA-16 are lower than those of LVA-8 and LVA-16. There are some other key results that can be observed from the figure also. Firstly, along each row (i.e., when T_{ph} is fixed), the cell for $T_{obs} = 9$ has the lowest ADE and FDE. This is not unexpected as these models were originally trained with $T_{obs} = 9$. Secondly, as one moves away from the $T_{obs} = 9$ column, the ADEs and FDEs of the two LVTA models increase at a slower rate than those of the LVA counterparts. This can be observed from the more drastic change of colour along the columns of the two LVA tables. Thirdly, if one compares the right bottom regions (where both T_{obs} and T_{ph} are large) of the ADE and FDE tables for LVA-8 and LVA-16, then one should notice that, for the cells corresponding to the same T_{obs} and T_{ph} values, LVA-8 has smaller ADEs than LVA-16 but it has larger FDEs than LVA-16. These results indicate that, toward the end of the trajectories, the predicted locations from LVA-8 deviate more from the ground truth than those predicted by LVA-16. However, with the help of the extra temporal attention in LVTA, no similar results are observed between the large (T_{obs}, T_{ph}) regions of LVTA-8 and LVTA-16. The above experiments confirm that the improvement in prediction of LVTA over LVA is significant. They also demonstrate the advantage of incorporating the temporal attention mechanism into the network architecture.

Figure 10 shows some examples of predicted trajectories from LVTA-8 and LVTA-16 on the Edinburgh dataset. On the top row, $T_{obs} = 9$ and $T_{ph} = 8$; on the bottom row, $T_{obs} = T_{ph} = 20$. As expected, when the two LVTA models generate trajectories with a smaller prediction horizon (top row), they perform better than cases where the prediction horizon is quite large (bottom row). Although the trajectories generated by the two models still follow the directions of the ground truth trajectories closely, deviations of the end points of the predicted and ground truth trajectories are noticeable in the last three cases in the bottom row of the figure.

F. COMPUTATION TIME

To analyze the computation time of LVA and LVTA, the vanilla LSTM prediction method is used as the baseline for comparison as it is the basic LSTM trajectory prediction method. All the methods were implemented under the same coding environment (Pytorch version 0.3.1 and Python version 3.6) and trained on the same desktop having a GTX-1080 GPU on the Central Station dataset. The results of the training time of these three methods are summarised in Table 4. The training time of LVA is more than twice the time of the vanilla LSTM. This is due to two LSTM layers (velocity layer and location layer) being used in LVA, compared to only one LSTM layer in the vanilla LSTM method. The location-velocity attention mechanism is also

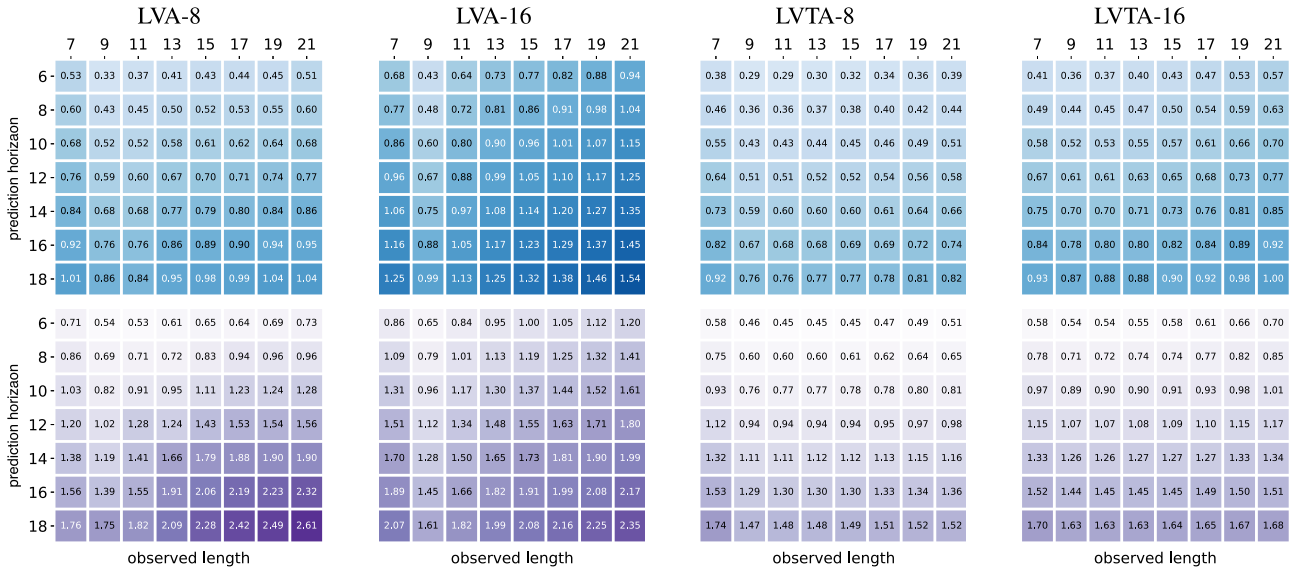


FIGURE 9. ADEs (top row) and FDEs (bottom row) of LVA-8, LVA-16, LVTA-8, and LVTA-16 under different observed lengths (T_{obs}) and different prediction horizons (T_{ph}) on the Edinburgh test set. All the models were trained on the Central Station training set. The lighter is the colour of a cell, the better is the performance.

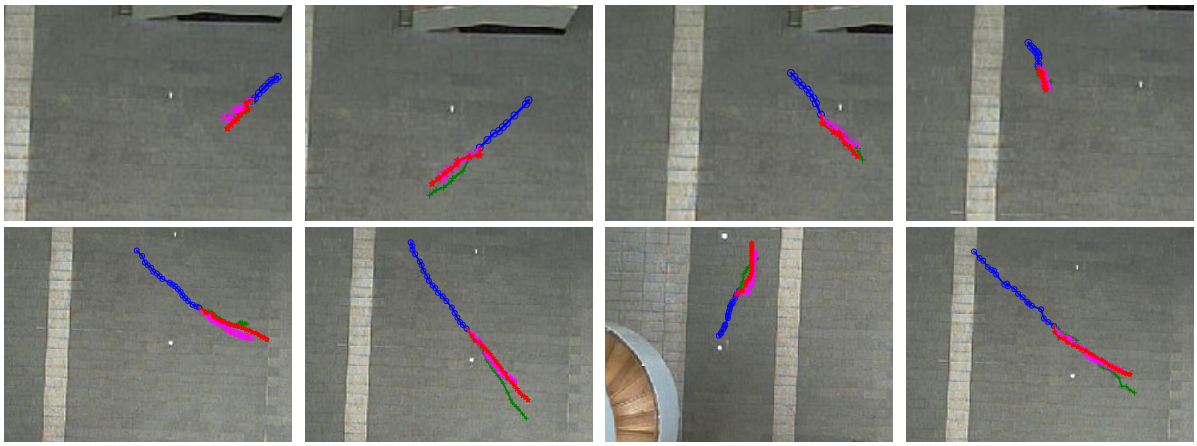


FIGURE 10. Prediction results on the Edinburgh dataset. Colour codes: blue: input observed trajectories; green: ground truth trajectories; pink: LVTA-8; red: LVTA-16. Top row: $T_{\text{obs}} = 9$, $T_{\text{ph}} = 8$; bottom row: $T_{\text{obs}} = 20$, $T_{\text{ph}} = 20$.

TABLE 4. Computation time and performance comparison.

	Training Time	ADE ↓	FDE ↓
Vanilla LSTM	1 ×	–	–
LVA	2.34 ×	31.78%	30.49%
LVTA	2.89 ×	36.93%	37.80%

responsible for the extra training time in LVA. Similar to LVA, LVTA also requires more than twice the training time of the vanilla LSTM. Compared to LVA, the additional training time required by LVTA mainly comes from the extra temporal attention mechanism.

The last two columns of Table 4 denote the reduction in ADE and FDE of LVA and LVTA compared to the baseline vanilla LSTM. The percentage values in these columns are computed as follows. Let ϵ_{van} be the ADE (similarly for the FDE) of the vanilla LSTM method and ϵ be the ADE of LVA (similarly for LVTA). The error reduction is defined as $(\epsilon_{\text{van}} - \epsilon)/\epsilon_{\text{van}} \times 100\%$. The Table shows a clear positive correlation between the training time and the improvement in trajectory prediction of the three methods, with LVTA achieving the smallest ADE and FDE.

V. CONCLUSION

We have presented a pedestrian trajectory prediction method that comprises a location LSTM layer, a velocity LSTM layer, and a tweak module which incorporates a joint location-velocity attention layer. Temporal attention mechanisms are used in the two LSTM layers. Experimental results demonstrate effectiveness of the temporal attention mechanism in our LVTA method, giving significant improvement to the prediction results than our previous LVA method. Compared to existing pedestrian trajectory prediction methods,

our LVTA method outperforms several methods on the Central Station and Edinburgh datasets and performs competitively against recent state-of-the-art methods on the complex ETH/UCY dataset. Furthermore, our method is a simpler method in that it does not require scene context information or trajectories of neighbouring pedestrians in the scene.

We have also thoroughly evaluated the generalizability of LVTA in terms of using different observed lengths and prediction lengths as well as applying a pretrained LVTA model to predict trajectories from a different scene. The results of these evaluations show that LVTA can yield good prediction results in such circumstances, even when the prediction length is different from that in the pretrained model. Compared to our previous LVA method, LVTA demonstrates even better generalizability.

In our proposed LVTA method, the two temporal attention mechanisms and location-velocity (LV) attention mechanism are separately handled: the temporal mechanisms manipulate the hidden states from the LSTMs to create context vectors, whereas the LV attention mechanism is a bit obscure and hidden inside the tweak module. One possible future extension is to jointly consider these two types of attentions by making some changes to the network structure of the prediction model. This may help improve the prediction performance further. Our other future research directions include incorporating LVTA into a trajectory analysis system that analyzes pedestrian trajectories for abnormal movement pattern detection, trajectory clustering, and trajectory counting for surveillance applications.

VI. ACKNOWLEDGMENT

The authors gratefully acknowledge the support from NVIDIA Corporation of a Titan Xp GPU used in this research.

REFERENCES

[1] K. Saleh, M. Hossny, and S. Nahavandi, "Contextual recurrent predictive model for long-term intent prediction of vulnerable road users," *IEEE Trans. Intell. Transp. Syst.*, to be published.

[2] W. Zhang, L. Sun, X. Wang, Z. Huang, and B. Li, "SEABIG: A deep learning-based method for location prediction in pedestrian semantic trajectories," *IEEE Access*, vol. 7, pp. 109054–109062, 2019.

[3] C. Wang, L. Ma, R. Li, T. S. Durrani, and H. Zhang, "Exploring trajectory prediction through machine learning methods," *IEEE Access*, vol. 7, pp. 101441–101452, 2019.

[4] B. Volz, H. Mielenz, I. Gilitschenski, R. Siegwart, and J. Nieto, "Inferring pedestrian motions at urban crosswalks," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 2, pp. 544–555, Feb. 2019.

[5] 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.

[6] S. Pellegrini, A. Ess, K. Schindler, and L. van Gool, "You'll never walk alone: Modeling social behavior for multi-target tracking," in *Proc. IEEE 12th Int. Conf. Comput. Vis.*, Sep. 2009, pp. 261–268.

[7] 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.

[8] 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.

[9] S. Kim, S. J. Guy, W. Liu, D. Wilkie, R. W. H. Lau, M. C. Lin, and D. Manocha, "BRVO: Predicting pedestrian trajectories using velocity-space reasoning," *Int. J. Robot. Res.*, vol. 34, no. 2, pp. 201–217, 2015.

[10] Y. Li, "Pedestrian path forecasting in crowd: A deep spatio-temporal perspective," in *Proc. ACM Multimedia Conf. (MM)*, 2017, pp. 235–243.

[11] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human trajectory prediction in crowded spaces," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 961–971.

[12] 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. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 336–345.

[13] 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," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1–7.

[14] N. Nikhil and B. T. Morris, "Convolutional neural network for trajectory prediction," in *Proc. ECCV Workshop*, Sep. 2018, pp. 1–11.

[15] N. Deo and M. M. Trivedi, "Convolutional social pooling for vehicle trajectory prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. Workshops (CVPRW)*, Jun. 2018, pp. 1468–1476.

[16] Y. Xu, Z. Piao, and S. Gao, "Encoding crowd interaction with deep neural network for pedestrian trajectory prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 5275–5284.

[17] I. Hasan, F. Setti, T. Tsesmelis, A. Del Bue, F. Galasso, and M. Cristani, "MX-LSTM: Mixing tracklets and vislets to jointly forecast trajectories and head poses," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 6067–6076.

[18] H. Xue, D. Q. Huynh, and M. Reynolds, "SS-LSTM: A hierarchical LSTM model for pedestrian trajectory prediction," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Mar. 2018, pp. 1186–1194.

[19] S. Haddad, M. Wu, H. Wei, and S. K. Lam, "Situation-aware pedestrian trajectory prediction with spatio-temporal attention model," in *Proc. 24th Comput. Vis. Winter Workshop*, 2019, pp. 4–13.

[20] P. Zhang, W. Ouyang, P. Zhang, J. Xue, and N. Zheng, "SR-LSTM: State refinement for LSTM towards pedestrian trajectory prediction," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 12085–12094.

[21] J. Liang, L. Jiang, J. C. Niebles, A. G. Hauptmann, and L. Fei-Fei, "Peeking into the future: Predicting future person activities and locations in videos," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 5725–5734.

[22] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially acceptable trajectories with generative adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2255–2264.

[23] A. Vemula, K. Muelling, and J. Oh, "Modeling cooperative navigation in dense human crowds," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2017, pp. 1685–1692.

[24] F. Bartoli, G. Lisanti, L. Ballan, and A. D. Bimbo, "Context-aware trajectory prediction," 2017, *arXiv:1705.02503*. [Online]. Available: http://arxiv.org/abs/1705.02503

[25] H. Zou, H. Su, S. Song, and J. Zhu, "Understanding human behaviors in crowds by imitating the decision-making process," in *Proc. AAAI*, 2018, pp. 1–9.

[26] C. Schöller, V. Aravantinos, F. Lay, and A. Knoll, "What the constant velocity model can teach us about pedestrian motion prediction," 2019, *arXiv:1903.07933*. [Online]. Available: http://arxiv.org/abs/1903.07933

[27] H. Xue, D. Huynh, and M. Reynolds, "Location-velocity attention for pedestrian trajectory prediction," in *Proc. IEEE Winter Conf. Appl. Comput. Vis. (WACV)*, Jan. 2019, pp. 2038–2047.

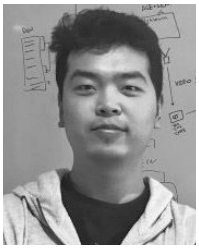
[28] K. Yamaguchi, A. C. Berg, L. E. Ortiz, and T. L. Berg, "Who are you with and where are you going?" in *Proc. CVPR*, Jun. 2011, pp. 1345–1352.

[29] H. Xue, D. Q. Huynh, and M. Reynolds, "Bi-prediction: Pedestrian trajectory prediction based on bidirectional LSTM classification," in *Proc. Int. Conf. Digit. Image Comput., Techn. Appl. (DICTA)*, Nov. 2017, pp. 307–314.

[30] A. Sadeghian, V. Kosaraju, A. Sadeghian, N. Hirose, H. Rezatofighi, and S. Savarese, "SoPhie: An attentive GAN for predicting paths compliant to social and physical constraints," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2019, pp. 1349–1358.

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

[32] S. Yi, H. Li, and X. Wang, "Pedestrian behavior understanding and prediction with deep neural networks," in *Proc. ECCV*. Cham, Switzerland: Springer, 2016, pp. 263–279.
[33] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. NIPS*, 2014, pp. 2672–2680.
[34] Y. Choi, M. Choi, M. Kim, J.-W. Ha, S. Kim, and J. Choo, "StarGAN: Unified generative adversarial networks for multi-domain image-to-image translation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8789–8797.
[35] O. Kupyn, V. Budzan, M. Mykhailych, D. Mishkin, and J. Matas, "DeblurGAN: Blind motion deblurring using conditional adversarial networks," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 8183–8192.
[36] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," in *Proc. ICLR*, 2015, pp. 1–15.
[37] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2015, pp. 1412–1421.
[38] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *Proc. ICML*, 2015, pp. 2048–2057.
[39] L. Yao, A. Torabi, K. Cho, N. Ballas, C. Pal, H. Larochelle, and A. Courville, "Describing videos by exploiting temporal structure," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Dec. 2015, pp. 4507–4515.
[40] 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. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1647–1656.
[41] J. Liu, G. Wang, L.-Y. Duan, K. Abdiyeva, and A. C. Kot, "Skeleton-based human action recognition with global context-aware attention LSTM networks," *IEEE Trans. Image Process.*, vol. 27, no. 4, pp. 1586–1599, Apr. 2018.
[42] S. Li, S. Bak, P. Carr, and X. Wang, "Diversity regularized spatiotemporal attention for video-based person re-identification," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 369–378.
[43] L. Wu, Y. Wang, X. Li, and J. Gao, "Deep attention-based spatially recursive networks for fine-grained visual recognition," *IEEE Trans. Cybern.*, vol. 49, no. 5, pp. 1791–1802, May 2019.
[44] M. Yang, W. Tu, J. Wang, F. Xu, and X. Chen, "Attention based LSTM for target dependent sentiment classification," in *Proc. AAAI*, 2017, pp. 1–2.
[45] F. Karim, S. Majumdar, H. Darabi, and S. Chen, "LSTM fully convolutional networks for time series classification," *IEEE Access*, vol. 6, pp. 1662–1669, 2018.
[46] A. Vemula, K. Muelling, and J. Oh, "Social attention: Modeling attention in human crowds," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1–7.
[47] 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.
[48] A. Sadeghian, F. Legros, M. Voisin, R. Vesel, A. Alahi, and S. Savarese, "CAR-Net: Clairvoyant attentive recurrent network," in *Proc. ECCV*, Sep. 2018, pp. 151–167.
[49] (Jun. 2018). *agrimgupta92/sgan*. [Online]. Available: https://github.com/agrimgupta92/sgan
[50] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. NIPS*, 2014, pp. 3104–3112.
[51] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1724–1734.
[52] B. Zhou, X. Wang, and X. Tang, "Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2871–2878.
[53] A. Lerner, Y. Chrysanthou, and D. Lischinski, "Crowds by example," *Comput. Graph. Forum*, vol. 26, no. 3, pp. 655–664, Sep. 2007.
[54] B. Majecka, "Statistical models of pedestrian behaviour in the forum," M.S. thesis, School Informat., Univ. Edinburgh, Edinburgh, U.K., 2009.
[55] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980



HAO XUE 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. 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 Australian 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 on 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., in 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.