Convolutional Neural Network for Trajectory Prediction

Nishant Nikhil[1,2] and Brendan Tran Morris[2][0000−0002−8592−8806]

[1] Indian Institute of Technology Kharagpur
nishantnikhil@iitkgp.ac.in
[2] University of Nevada, Las Vegas
brendan.morris@unlv.edu

Abstract. Predicting trajectories of pedestrians is quintessential for autonomous robots which share the same environment with humans. In order to effectively and safely interact with humans, trajectory prediction needs to be both precise and computationally efficient. In this work, we propose a convolutional neural network (CNN) based human trajectory prediction approach. Unlike more recent LSTM-based moles which attend sequentially to each frame, our model supports increased parallelism and effective temporal representation. The proposed compact CNN model is faster than the current approaches yet still yields competitive results.

Keywords: Convolutional Neural Network, Trajectory Prediction, Anticipating Human Behavior

1 Introduction

Autonomous robots like self-driving cars on a road or a food-delivery robot in a restaurant must share the same space with humans. In order to do so in a safe and acceptable manner, these robots must be able to understand and cooperate with humans. One task of paramount importance for avoiding collisions and for smooth maneuvering is to accurately predict the future trajectories of humans in their shared space. Further, given the wide diversity of platforms and environments for which prediction may be required (e.g. small robots with limited computing capabilities or without connectivity to cloud computing resources), simple models with better time complexity are desired.

Traditionally, hand-crafted features were used for trajectory prediction and modeling motion of pedestrians' trajectory with respect to others surrounding them. [1] propose a discrete choice framework for pedestrian dynamics, modeling short-term behavior of individuals as a response to the presence of other pedestrians. The Social Force model [2] incorporates two interactive forces for microsimulation of crowds. Attractive forces guiding the pedestrians towards their goal and repulsive forces for encouraging collision avoidance in-between the pedestrians and in-between a pedestrian and environmental obstacles. Yamaguchi et al. [3] solves the same problem as an energy minimization problem.

While successful, hand-crafted features are hard to scale since influencing factors must be described explicitly.

In recent years, Deep Neural Networks (DNN) have been utilized for the trajectory prediction task since they utilize a data-driven approach to tease out relationships and influences which may not have been apparent. These DNN-based approaches [4–7] have demonstrated impressive results. Almost all of these approaches are based on Recurrent Neural Networks (RNNs) [8] since a trajectory is a temporal sequence. As RNNs share parameters across time, they are capable of conditioning the model on all previous positions of a trajectory. Although theoretically, RNNs can retain information from all previous words of a sentence, practically they fail at handling long-term dependencies. Also, RNNs are prone to the vanishing and exploding gradient problems when dealing with long sequences.

Long Short-Term Memory (LSTM) networks [9], a special kind of RNN architecture, were designed to address these problems. Although LSTMs have been found to address the sequence based problems effectively but they need quite a bit of task-specific engineering like clipping gradients. Also in RNNs, predictions for later time-steps must wait for the predictions from preceding time-steps and hence can't be parallelized during training or inference time.

Recently, Convolutional Neural Network (CNN) based architectures have provided encouraging results in sequence-to-sequence tasks [10] like machine translation [11,12], image generation [13] and Image Captioning [14]. Inspired by these, we study CNNs for the task of trajectory prediction. This is the first work we are aware of to use an end-to-end convolutional architecture for trajectory prediction (Deo and Trivedi [15] used convolutional pooling for incorporating social context from hidden states of the LSTM network). We believe the CNN is superior to LSTM for temporal modeling since trajectories are continuous in nature, do not have complicated "state", and have high spatial and temporal correlation which can be exploited by computationally efficient convolution operations.

The major contribution of the work can be summarized as proposing a fast CNN-based model for trajectory prediction that is competitive with more complicated state-of-the-art LSTM-based techniques which require more contextual information. We discuss our CNN architecture in Section 2. Section 3 provides an experimental evaluation to highlight the efficacy of our approach and value due to simplicity. Finally, in Section 4, we conclude the paper with closing remarks.

2 Trajectory Prediction Method

Recent work in prediction has utilized recurrent networks to model temporal dependencies and sequence-like nature of trajectories using LSTMs. Most efforts in this area look to augment position input with social [4, 5] or scene [6, 16] context resulting in more complicated architectures. In contrast, our work seeks to simplify the network architecture and make more direct use of trajectory

structure (spatio-temporal consistency) by using highly efficient convolutions temporal support.

2.1 Problem Setup

For trajectory prediction, we are given the trajectory of all the pedestrians. It is assumed that each scene is pre-processed and we have the spatial coordinates of every i-th pedestrian at time t as $X_t = (x_t^i, y_t^i)$. That is, we have the pedestrian trajectory data as $X = \{X_1, X_2, X_3, X_4, \ldots, X_n\}$ for time steps $t = 1, 2, \ldots, T_{obs}$. Note: for simiplicity the pedestrian superscript i is not listed. We have to predict the future trajectories of all the pedestrians for time steps $t = T_{obs+1}, \ldots, T_{pred}$ as $\hat{Y} = \{\hat{Y}_1, \hat{Y}_2, \hat{Y}_3, \hat{Y}_4, \ldots, \hat{Y}_m\}$ all at once.

2.2 LSTM-Based Frameworks

Most current reserach in trajectory prediction has utilized LSTM cells for handling temporal dependencies. The working of LSTM cells are governed by the following equations:

$$f_t = \sigma_g(W_f X_t + U_f h_{t-1} + b_f) \tag{1}$$

$$i_t = \sigma_g(W_i X_t + U_i h_{t-1} + b_i) \tag{2}$$

$$o_t = \sigma_g(W_o X_t + U_o h_{t-1} + b_o) \tag{3}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \sigma_c(W_c x_t + U_c h_{t-1} + b_c) \tag{4}$$

$$h_t = o_t \circ \sigma_h(c_t) \tag{5}$$

In these equations, X_t is the input vector to the LSTM unit, f_t is the forget gate's activation vector, i_t is the input gate's activation vector, o_t is the output gate's activation vector, h_t is the output vector of the LSTM unit and c_t is the cell state vector. w, u, B are the parameters of weight matrices and bias vectors which are learned during the training.

The basic LSTM formulation has been extended to add more complicated LSTM units by adding more contextual information such as social cues (influence of neighboring humans) [4, 5] or environmental cues (influence of scene) [6, 17]. While these have been effective in improving prediction performance, they still utilize the LSTM which has hidden state h_t dependent on previous time-steps and can not be parallelized. Sequential evaluation limits the speed of any LSTM-based architecture.

2.3 CNN-Based Framework

In contrast with LSTM-based networks, our proposed network (Fig. 1) utilizes highly parallelizable convolutional layers to handle temporal dependencies. The CNN-network is actually a simple sequence-to-sequence architecture. Trajectory histories are used as input and embedded to a fixed size through a fully connected layer. Convolutional layers are stacked and used to enforce temporal consistency. Finally, the features from the final convolutional layer are concatenated

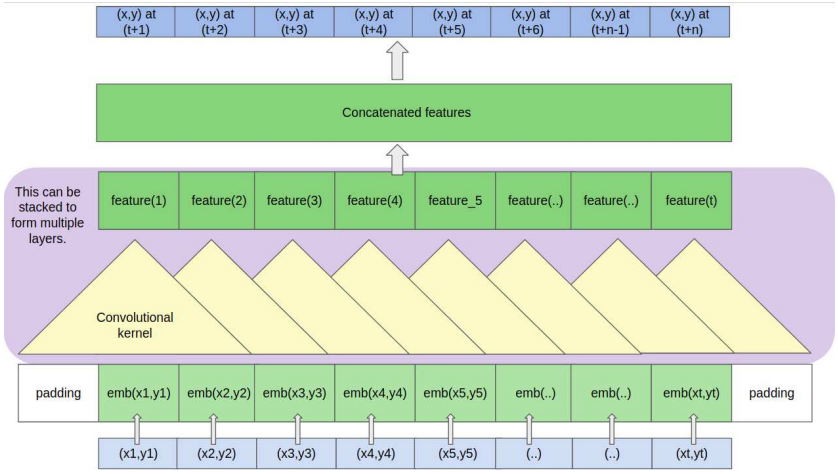

Fig. 1. Our convolutional model for trajectory prediction. Note that all operations are feed-forward in nature and hence can be parallelized.

and passed through a fully connected layer to generate all predicted positions $(x^t, y^t)_{t=t+1}^{t+t_{pred}}$ at once.

The model is inspired by the work of [14] which has to predict a discrete output for the neural machine translation task. In that setting, the output at the next time step is highly dependent on the current time step for grammatical coherency and CNNs performed well. The major differences between this work and theirs are that trajectory prediction provides continuous output rather than discrete items and our architecture predicts all future time steps at once. We constantly pad the input to convolution layer such that output from the convolutional layer is of the same size as input to the layer. This way, we can build a neural network as deep as we want. We build the network deep enough to capture the context from every time step of the observed trajectory. We discuss more in the next subsection.

Through an ablation study (Section 3.2), we found that predicting one time step at a time leads to worse results than all future times at once. We believe this is due to error of the current prediction being propagated forward in time in a highly correlated fashion. Also, unlike LSTM-based architectures which utilize a recurrent function to compute sequentially, all the computation in the proposed model are feed-forward in nature resulting in a significant performance boost with respect to inference time. Additionally, the convolutions can be easily parallelized.

2.4 Implementation Details

We use a kernel size of 3 for all the kernels, by ablation study we found that it works better than other odd kernel sizes when we apply symmetric padding. As

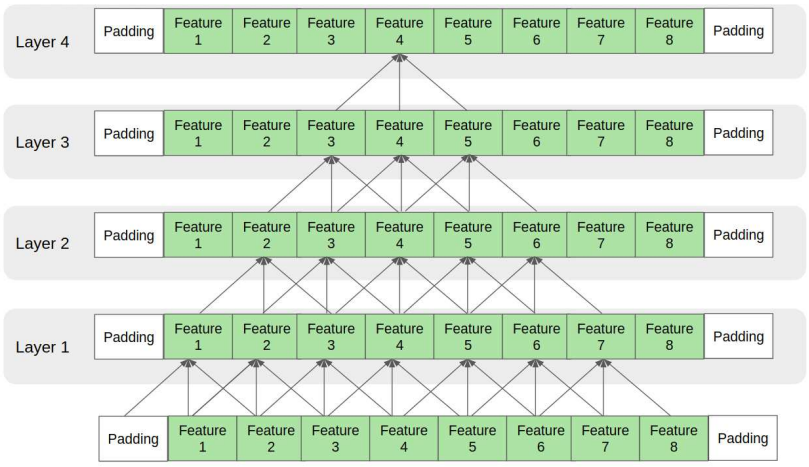

Fig. 2. For an input having eight temporal dimension and convolutional layers having kernel size of three, we need at least four layer to capture the context from all time-steps.

the observed trajectory length is eight for all the experiments we conduct, we use a four-layered convolutional network. As shown in Fig. 2, All the features in layer 4 capture context from all eight trajectory observations. Unlike temporal convolutional networks (TCNs), we do not use dilated convolutions because we do not want to lose information on such a small temporal dimension. Additionally, we use full rather than causal kernels since the output is a prediction. The embedding layer which converts the geometrical coordinates to embeddings has a dimension of 32, and the subsequent convolution layers produce outputs of the same dimensions. For optimization, we use Adam [18] with a learning rate of 0.001. We use a batch size of 32. The model is trained until the validation loss (L2 loss) stops decreasing.

3 Experiments

Following common practice in literature (e.g. [4]), experimental evaluation is conducted on publically available pedestrian trajectory datasets. Evaluation utilizes eight historical samples (3.2 s) to give a long-term prediction of the next 12 samples (4.8 s).

3.1 Datasets and Evaluation Criteria

Two publicly available datasets which provide over 1500 pedestrian trajectories in varied crowd settings are utilized in our experiments. The ETH dataset [19] consists of the ETH and HOTEL scenes while the UCY dataset [20] has the UNIV, ZARA1, and ZARA2 scenes. The trajectories are rich with challenging

human-human interaction scenarios such as group behavior, non-linear trajectories, people crossing paths, collision avoidance, and group formation and dispersion. All trajectory data has been converted from image to real-world coordinates and interpolated at 2.5 Hz.

As with prior work [5] [21], we use two metrics for computing prediction error:

1. Average Displacement Error (ADE): Computes the mean of euclidean distance between the points in predicted trajectory and the corresponding points in ground truth for all predicted time steps.

$$ADE = \frac{\sum_{t=obs+1}^{T_{pred}} \left\| Y_t - \hat{Y}_t \right\|}{T_{pred} - T_{obs}}$$

2. Final Displacement Error (FDE): The Euclidean distance between final destination as per the ground truth and the predicted destination at end of the prediction period T_{pred}.

$$FDE = \left\| Y_{T_{pred}} - \hat{Y}_{T_{pred}} \right\|$$

Similar to [4] [5], we follow leave-one-out approach. We train on four of the five crowd scenes and test on the remaining set. The trajectory is observed for 8-time steps (3.2 seconds), then the model makes the prediction for 12-time steps (4.8 seconds).

Table 1. Quantitative ADE/FDE for the task of predicting 12 future time steps given 8 previous time steps. More contextual information is provided from left to right ($^{+}$social, $^{++}$raw scene image)

Dataset	Ours	LSTM	S-GAN^{+}	S-LSTM^{+}	S-GAN-P^{+}	SoPhie^{++}
ETH	1.04/2.07	1.09/2.41	0.81/1.52	1.09/2.35	0.87/1.62	**0.70/1.43**
HOTEL	**0.59/1.17**	0.86/1.91	0.72/1.61	0.79/1.76	0.67/1.37	0.76/1.67
UNIV	0.57/1.21	0.61/1.31	0.60/1.26	0.67/1.40	0.76/1.52	**0.54/1.24**
ZARA1	0.43/0.90	0.41/0.88	0.34/0.69	0.47/1.00	0.35/0.68	**0.30/0.63**
ZARA2	**0.34/0.75**	0.52/1.11	0.42/0.84	0.56/1.17	0.42/0.84	0.38/0.78
AVG	0.59/1.22	0.70/1.52	0.58/1.18	0.72/1.54	0.61/1.21	**0.54/1.15**

3.2 Quantitative Evaluation

In Table 1, we compare prediction results against five different architectures:

1. LSTM: A simple Long Short-Term Memory architecture without any pooling mechanism, i.e. it doesn't consider any social context.

2. S-LSTM [5]: This model combines LSTMs with a social pooling mechanism to provide social context in a fixed rectangular grid.
3. S-GAN [4]: This model uses LSTM and variable social max-pooling mechanism in a Generative Adversarial Network (GAN) architecture to generate multiple plausible trajectories. The S-GAN-P variant also uses a social pooling mechanism.
4. SoPhie [6]: Apart from having LSTM and pooling for features in a GAN setting, this model applies a scene attention mechanism over the features extracted from images of the scene to augment trajectory information.

The results are organized by increasing contextual information (e.g. social pooling or raw images for scene information) from left to right. Note: that LSTM, S-LSTM, and S-GAN results in Table 1 were reported in [4].

We find that our model consistently outperforms the LSTM baseline even though they are utilizing the same basic position inputs. We speculate that this is because the CNNs do a better job at handling long-term dependencies than the LSTM specifically for continuous numerical regression where the notion of state is not complicated. Interestingly, ours is the best performing architecture for the HOTEL scene even without the use of social or environmental cues. This is likely due to the simplicity of the scene since even a simple linear regressor provides better results (0.39/0.72) than all reported here [4]. However, the simple CNN still performs very well even in more complicated scenarios (UNIV and ZARA2) and actually beats techniques that utilize social context. The UNIV result is most surprising since it is the most complicated scene with large crowds of people. In these situations, social context may not be relevant (Fig. 4(b)). In fact, the average performance is quite similar to S-GAN (provides many plausible trajectories), better than S-GAN-P (multiple trajectories with social pooling), and competitive with SoPhie even though those techniques use social context and scene image context (in the case of SoPhie).

Table 2. Speed comparison with other architectures

	LSTM	**S-GAN**	**S-GAN-P**	**Ours**
Time (s)	0.009	0.022	0.067	0.002
Speed-Up	7.44×	3.0×	1×	33.5×

The main advantage of our proposed CNN prediction architecture is the computational efficiency of convolution operations which can be highly parallelized. A speed comparison is provided in Table 2 which reports inference time in seconds and speed up factor with respect to the baseline S-GAN-P. The high speed of our method makes it well suited for mobile robot applications which need to make predictions in real-time.

Furthermore, to decide the number of layers we trained our architecture with different numbers of convolutional layers. Table 3 indicates that four layers performed the best. We believe this happens because three-layered networks are

Table 3. CNN layer ablation study

Layers	Three	Four	Five
ADE	0.60	0.58	0.70
FDE	1.30	1.20	1.40

not able to capture context from all time-steps and five-layered networks are over-parametrized.

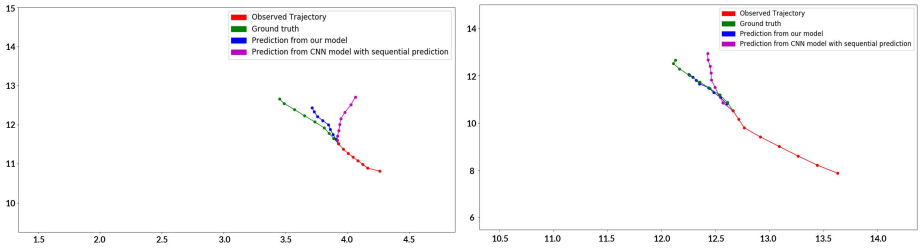

Fig. 3. Multi vs. Sequential Output. Trajectory prediction sequentially point-by-point performs poorly due to error propogation to future time-steps (trajectory curves off). Our multi-output model tends to be more resistant to such error accumulation.

3.3 Qualitative Evaluation

In Fig. 3 and Fig. 4, we examine the quality of trajectories produced by the CNN architecture. One important finding was that sequential prediction (similar to LSTM-based models) performed very poorly (Fig. 3). Prediction error for the maroon curve was propagated forward resulting in trajectories that "curved off" over time. In contrast, the multi-output CNN architecture was more resistant to this type of error accumulation.

Fig. 4 provides a comparison between the CNN (blue) and S-GAN (maroon). (a) provides an exmample when the CNN has a better prediction than S-GAN. In (b), S-GAN's social pooling causes poor prediction since it thinks all five pedestrians should be moving as a group. Their prediction of the two right moving pedestrians is strongly pulled to the left resulting in large error. In contrast, the CNN is able to independently predict with better results. The UNIV scene in particular is quite dense making the pooling operation challenging. In (c), both CNN and S-GAN fail as seen in the center. In particular, three pedestrians seem to move in unison to avoid something in the scene and therefore neither algorithm is aware. Finally, (d) shows an example of S-GAN performing better than CNN. It is interesting to note that for both (a) and (d), neither technique is actually working that well. Also, it is difficult to fully understand

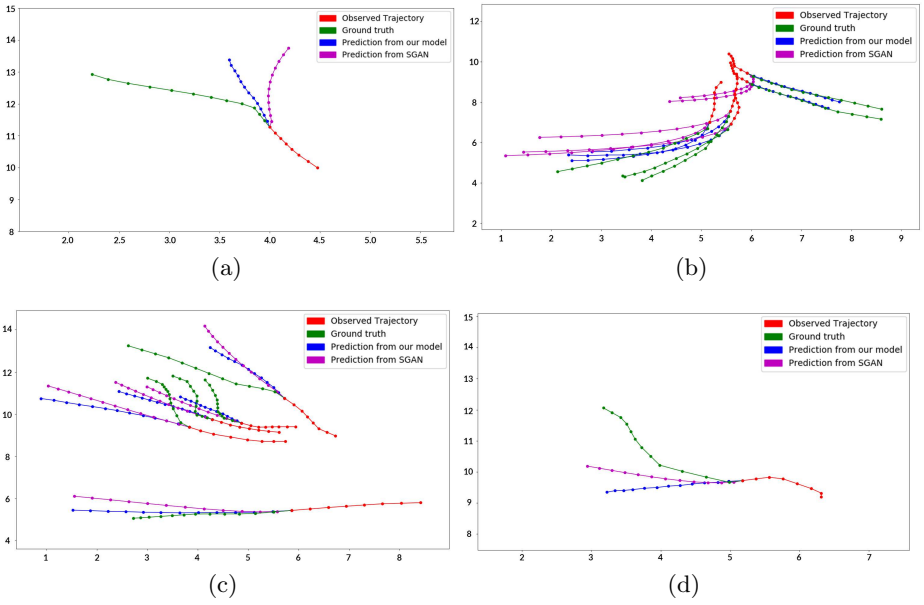

Fig. 4. Qualitative Comparison on UNIV. (a) CNN model is better able to interpolate while the S-GAN model seems to accumulate error in subsequent time-steps. (b) Social pooling erroneously combines all five pedestrians and thinks they all should be moving left. Without pooling, the CNN model is able to better predict the two pedestrians moving right. (c) Both models do a poor job of prediction, especially in the center. (d) SGAN provides a better prediction than CNN.

what is happening without overlaying the trajectories on the image frame. This strongly hints that trajectories alone (even with social pooling) is not sufficient to make robust prediction.

Note that unlike state-of-the-art architectures (e.g. S-GAN and SoPhie), our CNN prediction architecture does not include any context outside of individual trajectory information. Similar social pooling schemes could be added and further improvements are expected. Additionally, S-GAN reported a 49× speed up over S-LSTM which would make our CNN architecture 500× S-LSTM.

4 Conclusions

We present a convolutional architecture based neural network model for trajectory prediction. The simple model gives competitive results with the current state-of-art LSTM-based models while providing better inference time performance. We hope that following this work, more people would be interested in utilizing clever convolutional architectures for trajectory prediction. Given the current architecture is quite simple, future work will examine the use of dilated convolutions to decrease the number of layers while maintaining the same receptive field and incorporating social context into the model.

References

1. Antonini, G., Bierlaire, M., Weber, M.: Discrete choice models of pedestrian walking behavior. Transportation Research Part B: Methodological **40**(8) (2006) 667–687
2. Helbing, D., Molnar, P.: Social force model for pedestrian dynamics. **51** (05 1998)
3. Yamaguchi, K., Berg, A.C., Ortiz, L.E., Berg, T.L.: Who are you with and where are you going? In: CVPR 2011. (June 2011) 1345–1352
4. Gupta, A., Johnson, J., Fei-Fei, L., Savarese, S., Alahi, A.: Social gan: Socially acceptable trajectories with generative adversarial networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Number CONF (2018)
5. Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., Savarese, S.: Social lstm: Human trajectory prediction in crowded spaces. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (June 2016)
6. Sadeghian, A., Kosaraju, V., Sadeghian, A., Hirose, N., Savarese, S.: Sophie: An attentive gan for predicting paths compliant to social and physical constraints. arXiv preprint arXiv:1806.01482 (2018)
7. Fernando, T., Denman, S., Sridharan, S., Fookes, C.: Soft + hardwired attention: An LSTM framework for human trajectory prediction and abnormal event detection. CoRR **abs/1702.05552** (2017)
8. Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., Khudanpur, S.: Recurrent neural network based language model. In: Eleventh Annual Conference of the International Speech Communication Association. (2010)
9. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8) (November 1997) 1735–1780
10. Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. CoRR **abs/1803.01271** (2018)

11. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning. CoRR **abs/1705.03122** (2017)
12. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. CoRR **abs/1706.03762** (2017)
13. Oord, A.v.d., Kalchbrenner, N., Vinyals, O., Espeholt, L., Graves, A., Kavukcuoglu, K.: Conditional image generation with pixelcnn decoders. In: Proceedings of the 30th International Conference on Neural Information Processing Systems. NIPS'16, USA, Curran Associates Inc. (2016) 4797–4805
14. Aneja, J., Deshpande, A., Schwing, A.: Convolutional image captioning. In: Computer Vision and Pattern Recognition. (2018)
15. Deo, N., Trivedi, M.M.: Convolutional social pooling for vehicle trajectory prediction. CoRR **abs/1805.06771** (2018)
16. Xue, H., Huynh, D.Q., Reynolds, M.: Ss-lstm: A hierarchical lstm model for pedestrian trajectory prediction. In: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV). (March 2018) 1186–1194
17. Xue, H., Huynh, D.Q., Reynolds, M.: Ss-lstm: A hierarchical lstm model for pedestrian trajectory prediction. In: 2018 IEEE Winter Conference on Applications of Computer Vision (WACV). (March 2018) 1186–1194
18. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014)
19. Pellegrini, S., Ess, A., Van Gool, L.: Improving data association by joint modeling of pedestrian trajectories and groupings. In: Proceedings of the 11th European Conference on Computer Vision: Part I. ECCV'10, Berlin, Heidelberg, Springer-Verlag (2010) 452–465
20. Leal-Taix, L., Fenzi, M., Kuznetsova, A., Rosenhahn, B., Savarese, S.: Learning an image-based motion context for multiple people tracking. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014)
21. Lee, N., Choi, W., Vernaza, P., Choy, C.B., Torr, P.H.S., Chandraker, M.K.: Desire: Distant future prediction in dynamic scenes with interacting agents. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017) 2165–2174