Asymmetrical Bi-RNN for Pedestrian Trajectory Encoding

Raphaël Rozenberg, Fabien Moutarde, Joseph Gesnouin

Asymmetrical Bi-RNN for Pedestrian Trajectory Encoding

Raphaël Rozenberg[1,2], Joseph Gesnouin[2,3], and Fabien Moutarde[2]

[1]École Normale Supérieure - PSL, 75005 Paris, France, first.last@ens.psl.eu
[2]Centre de Robotique, MINES ParisTech - PSL, 75006 Paris, France, first.last@mines-paristech.fr
[3]Institut Vedecom, 78000 Versailles, France, first.last@vedecom.fr

***Abstract*— Pedestrian motion behavior involves a combination of individual goals and social interactions with other agents. In this article, we present an asymmetrical bidirectional recurrent neural network architecture called U-RNN to encode pedestrian trajectories and evaluate its relevance to replace LSTMs for various forecasting models. Experimental results on the Trajnet++ benchmark show that the U-LSTM variant yields better results regarding every available metrics (ADE, FDE, Collision rate) than common trajectory encoders for a variety of approaches and interaction modules, suggesting that the proposed approach is a viable alternative to the *de facto* sequence encoding RNNs. Our implementation of the asymmetrical Bi-RNNs for the Trajnet++ benchmark is available at: github.com/JosephGesnouin/Asymmetrical-Bi-RNNs-to-encode-pedestrian-trajectories.**

***Index Terms*—Sequence Encoding, Pedestrian Safety, Trajectory Forecasting**

I. Introduction

A. Pedestrian trajectory forecasting

Pedestrian trajectory prediction from past positions using social interactions has been steadily receiving attention by the research community, as it plays a crucial role in various applications leading to the deployment of intelligent transport systems [1], [2].

Following the success of Social LSTM [3] in trajectory forecasting in crowded scenes, a variety of approaches has been proposed that focused on efficiently leveraging social interactions from a scene [4]–[9]. In this article, we elude the question of improving social interactions models, and focus on the encoding of the trajectories of individual pedestrians by using U-RNNs (our asymmetrical Bi-RNNs) instead of regular LSTMs.

Using the recent Trajnet++ benchmark [5] and with respect to various available learning architectures that forecast pedestrians trajectories, we evaluate the effectiveness of U-RNNs for efficient pedestrian trajectories encoding. We then provide insight into designing improved motion encoders prior to the application of interaction modules for the task of pedestrian trajectory prediction.

B. From Bi-RNNs to U-RNNs

U-RNN is a bidirectional recurrent neural network architecture that was informally introduced in [10] under the form of U-GRUs for Knowledge Tracing. The objective of this work is to investigate whether U-RNNs could replace regular RNNs or Bi-RNNs for trajectory encoding.

Bi-RNNs [11] address a drawback of Recurrent Neural Networks (RNNs), which is that they cannot take the future into account when they encode an input, which may be desirable [12] for some cases. For example, in the case of pedestrian trajectory prediction [13], [14], one could expect that some movements are influenced by anticipation of a potential obstacle. Bi-RNNs produce two outputs, one that is obtained by reading the input forward and one by reading the input backwards. Concatenation or some other operation is then applied.

However, an aspect of Bi-RNNs that could be undesirable is the architecture's symmetry in both time directions. Bi-RNNs are often used in natural language processing, where the order of the words is almost exclusively determined by grammatical rules and not by temporal sequentiality. However, in trajectory prediction, the data has a preferred direction in time: the forward direction. Another potential drawback of Bi-RNNs is that their output is simply the concatenation of two naive readings of the input in both directions. In consequence, Bi-RNNs never actually read an input by knowing what happens in the future. Conversely, the idea behind U-RNN, illustrated in Fig. 1, is to first do a backward pass, and then use during the forward pass information about the future. By using an asymmetrical Bi-RNN to encode pedestrian trajectories, we accumulate information while knowing which part of the information will be useful in the future as it should be relevant to do so if the forward direction is the preferred direction of the data.

II. Related work

A. Encoder-Interaction-Decoder pipeline

The most common pipeline for pedestrian trajectory prediction consists of:

1) A **sequence encoder** for the past coordinates of each pedestrian independently. The encoder is usually a RNN, such as a LSTM.
2) An **interaction module** for taking into account the neighbors trajectories. The most common way to take into account the effect of interactions between agents in their trajectories is to decode the past positions while pooling on a spatial grid with either the neighbors' positions (*i.e, occupancy pooling*), their relative velocities (*i.e, directional pooling*) [5], or their RNN hidden states [3]. This last approach is very popular and is known under the name of *social pooling*.
3) A **decoder** that predicts future coordinates. A common approach is to use a RNN for decoding. Some authors

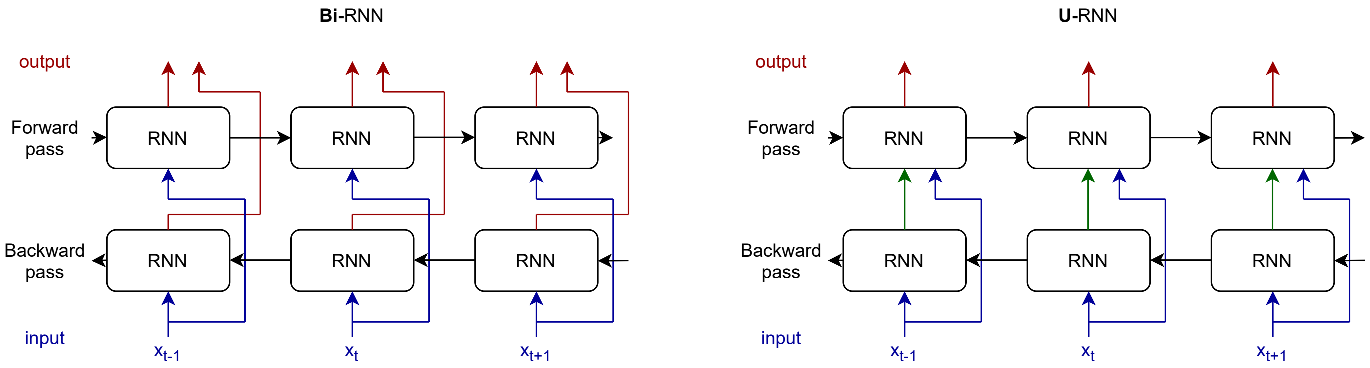

Fig. 1. Comparison between Bi-RNN and U-RNN architectures (blue: inputs - red: outputs - black: hidden states - green: intermediate output). U-RNN can use the information from the future during the forward pass, whereas the Bi-RNN only concatenates two naive readings in both directions.

found that this can lead to error accumulation, and that a simple multi-layer perceptron (MLP) that predicts simultaneously all future positions performs better [15]. However, taking into account interactions between pedestrians requires to predict the coordinates one step at a time, so RNNs are generally preferred.

Most of past years research focused on improving the interaction module, with only limited new methods since *Social-LSTM* [3], or in developing approaches that take inspiration in popular frameworks such as Transformers [16] or contrastive learning [17] in order to deter the model from predicting colliding or too uncomfortable trajectories. However, little work has been published on the influence of the encoder and thus on the importance of past coordinates, even if it would be easily applicable on all models that use this pipeline.

B. Alternative approaches.

a) Learning-free algorithms.: The straight line at constant speed using the last known velocity is a reasonable approximation for the problem at hand [18], given that we only try to predict the next few seconds. More complex learning-free methods can also be successfully applied, some generic, such as the Kalman Filter, and some specific, such as Optimal Reciprocal Collision Avoidance (ORCA) [19], which ensures that trajectories do not collide, which is not necessarily the case with other methods, especially the straight line.

b) Other methods.: Even though non-RNN methods cannot take advantage of the research on interaction modules, alternative machine learning approaches have been developed. Convolutional Neural Networks are faster than RNN-based methods due to parallelization, but the performances are significantly lower [20]. Some authors have explored the popular Transformers architecture, but the results are inferior to those of RNNs with state-of-the-art social interaction modules [16]. Research has also been conducted on applying Inverse Reinforcement Learning (IRL) to the pedestrian trajectory prediction problem [21], even though retrieving the pedestrian cost function requires much more computation than learning a predictor.

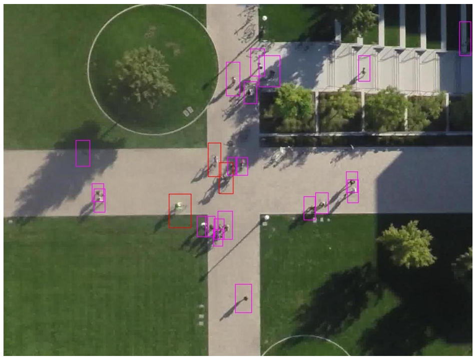

Fig. 2. Sample from the Stanford Drone Dataset (which is not included in the Trajnet++ benchmark). The environment would play an important role in order to predict trajectories that do not go on the lawn.

C. What information is relevant?

a) Scene context as an additional modality.: The Trajnet++ dataset does not include the pedestrians' environment, but some argue that it is sometimes necessary in order to predict trajectories correctly [15]. Indeed, in situations such as the one in Fig. 2, it would be very difficult to predict plausible trajectories since the environment would play an important role in order to predict trajectories that do not go on the lawn. However, the environment's additional information seems to make generalization more difficult [18].

b) Neighbors past coordinates.: Most methods make use of neighbors past and present positions. However, it seems that knowing even future neighbors positions is useless in terms of prediction error [18]. Indeed, global trajectories are not that much affected by interactions. Still, neglecting the influence of neighbors inevitably leads to collisions: relevant metrics for pedestrian trajectory prediction take this into account in addition to purely spatial errors, in order to produce physically feasible trajectories.

III. Methodology

We based our experiments on the Trajnet++ LSTM baseline [5] with respect to a variety of interaction modules: *directional*, *occupancy* and *social* pooling. All hyper-parameters except for the encoder remained unchanged. For clarification

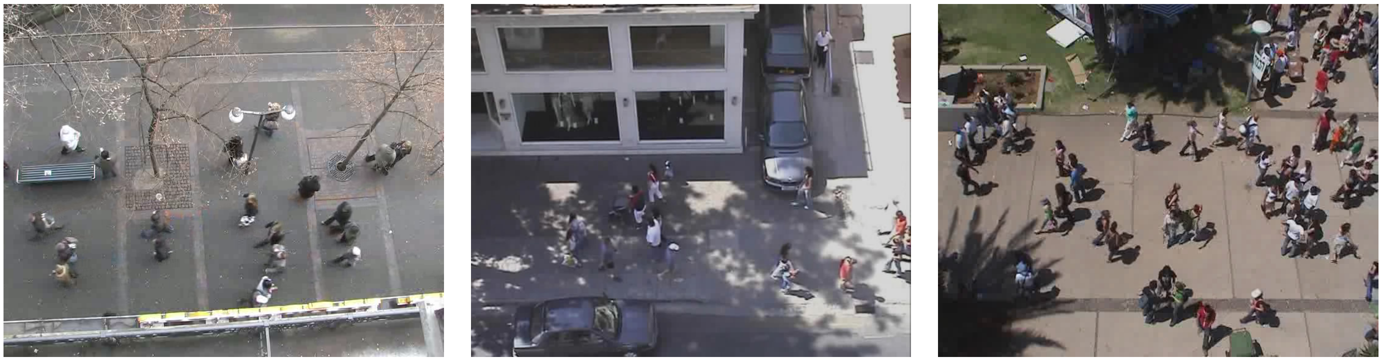

Fig. 3. Images from different datasets from which the Trajnet++ benchmark trajectories are extracted. Left: ETH-hotel dataset - Center: UCY-zara dataset - Right: UCY-students dataset.

purposes, we further explain our methodology for the *directional* pooling case.

A. Input embedding

The input data consists of coordinates $(x_t)_{t\in[\![1,T_{obs}]\!]}$ for each pedestrian. In order to allow easier generalization, we use velocities $(v_t)_{t\in[\![1,T_{obs}-1]\!]}$ instead with $v_t = x_{t+1} - x_t$.

From the trajectory velocities $(v_t)_t$ of a single pedestrian, we obtain the trajectory embeddings $(e_t)_{t\in[\![1,T_{obs}-1]\!]}$ with

$$e_t = f(v_t, W_e)$$

where f is a single-layer perceptron, and W_e are learnable weights that are shared among pedestrians.

B. U-RNN architecture

The backward and forward hidden states $(h_t^b)_{t\in[\![1,T_{obs}-1]\!]}$ and $(h_t^f)_{t\in[\![1,T_{obs}-1]\!]}$ are obtained according to these equations:

$$\begin{aligned} h_{t-1}^b &= RNN(h_t^b, e_t, W_b) \\ h_{t+1}^f &= RNN(h_t^f, [e_t, h_t^b], W_f) \end{aligned}$$

where W_b and W_f are learnable weights that are shared among pedestrians, and $[\cdot,\cdot]$ denotes concatenation.

The last hidden state $h_{T_{obs}}^f$ is then used as the encoding of the sequence.

C. Decoder

For decoding, we used a RNN and directional pooling, with a learnable $GridPooling$ function that involves average pooling and a linear embedding, all of which we do not detail here and was implemented in [5]. The predicted positions $(o_t^i)_{t\in[\![1,T_{pred}]\!]}$ of pedestrian i are obtained according to these equations:

$$\begin{aligned} h_1^i &= h_{T_{obs}}^{f,i} \\ e_t^i &= f(v_t^i, W_e) \\ I_t^i &= GridPooling(v_t^{-i}) \\ h_{t+1}^i &= RNN(h_t^i, [e_t^i, I_t^i], W_d) \\ o_t^i &= g(h_t^i, W_{out}) \end{aligned}$$

where $(v_t^i)_t$, $(e_t^i)_t$, $(I_t^i)_t$, $(h_t^i)_t$ are respectively the velocities, velocity embeddings, interaction embeddings and decoder hidden states for pedestrian i, W_e, W_d and W_{out} are learnable weights that are shared among pedestrians (W_e being the same as for the encoder), v^{-i} denotes velocities of pedestrians other than i and $[\cdot,\cdot]$ denotes concatenation.

IV. Experiments

A. The Trajnet++ benchmark

There are several datasets that are aimed at evaluating pedestrian motion prediction, with very diverse characteristics [22]. We chose the Trajnet++ benchmark [5], that aggregates several common pedestrian trajectories datasets, emphasis the importance of quantifying the physical feasibility of a model prediction and only evaluates on trajectories where there are interactions between pedestrians.

Data. Trajnet++ data consists of trajectories that have been extracted from real-life videos and that are under the form of spatial coordinates. The framerate is 2.5 frames per second. The datasets that are used are:

- ETH [23], itself subdivided into ETH-hotel and ETH-uni. ∼650 tracks extracted from 25 min of video.
- UCY [24], itself subdivided into UCY-zara and UCY-students. ∼700 tracks extracted from 16 min of video.
- WildTrack [25], ∼650 tracks extracted from an hour of video.
- L-CAS [26], ∼1100 tracks extracted from 49 min of video.
- CFF [27], Large-scale dataset of ∼42 million trajectories extracted from real-world train stations.

Fig. 3 illustrates ETH and UCY datasets with sample images from videos from which the spatial coordinates were extracted.

In addition, synthetic data generated using ORCA [19] is also used.

Task. The goal is to predict the spatial coordinates of pedestrians in the near future (12 frames, i.e. 4.8 seconds), using only the near past (9 frames, i.e. 3.6 seconds). In each scene (set of different agents' trajectories over a given duration), a primary pedestrian is designated for evaluation purposes.

Categories. The scenes in the data are subdivided into categories with respect to the primary pedestrian of the scene illustrates. Type I and Type II denote respectively static primary pedestrian trajectories and trajectories that are correctly predicted with an extended Kalman filter. Type III is the benchmark's type of interest, as it regroups all scenes where the primary pedestrian has interactions with other agents. Type IV is used for the remaining scenes, where the primary pedestrian trajectory seems unpredictable even when given the social environment.

In addition to the four main types, Type III is further subdivided into four categories that describe the main type of interaction that is occurring: Leader-follower (the primary pedestrian follows someone else), Collision avoidance (the primary pedestrian had to avoid someone else), Group (the primary pedestrian is part of a group) and Others.

Metrics. There are four main metrics. Two are spatial errors: Average Displacement Error (ADE) and Final Displacement Error (FDE), that are expressed in meters. The other two are collision errors: Prediction Collision (Col-I) and Ground Truth Collision (Col-II), that are expressed in percentage. Col-I is the fraction of collisions between the primary pedestrian predicted trajectory and the other pedestrians predicted trajectories, and thus represents how physically realistic the predicted scene is, regardless of reality. Col-II, on the other hand, is the fraction of collisions between the primary pedestrian predicted trajectory and the other pedestrians *real* trajectories. Therefore, it represents how physically realistic the predictions are individually.

Evaluation. According to the Trajnet++ benchmark, the performance is evaluated on ∼3000 scenes from ETH and UCY datasets, as well as on ∼4000 synthetic scenes. The benchmark gives metrics for each type and sub-type of scene. The score that is chosen in order to compare models on the public leaderboard is FDE computed on Type III (Interacting) scenes from the real datasets (∼1700 scenes), with Col-I as the secondary score (computed on the same data). Until the end of March 2021, the secondary score was FDE computed on Type III scenes from the synthetic dataset, but it was abandoned because predicting synthetic trajectories had become a solved problem. On the contrary, while performances seem to have reached a limit with respect to FDE (more than one meter on a 4.8 seconds horizon), the current challenge is to be able to predict physically feasible scenes while keeping a good FDE.

TABLE I

RESULTS FOR SEVERAL BASELINES AND FOR THE BEST SUBMISSION ON THE TRAJNET++ PUBLIC LEADERBOARD (WITH RESPECT TO FDE).

Model	ADE (m)	FDE (m)	Col-I (%)	Col-II (%)
Kalman filter	0.87	1.69	**0**	19.5
Constant velocity [18]	0.68	1.42	14.3	15.2
ORCA [19]	0.72	1.42	**0**	**11.3**
Vanilla LSTM	0.67	1.43	15.2	12.3
AMENet [28]	0.62	1.30	14.1	16.9
AIN [29]	0.62	1.24	10.7	17.1
PecNet [30]	0.57	1.18	15.0	14.3
Social NCE [17]	**0.53**	**1.14**	5.3	**11.3**

B. Baselines

We used the following baselines for comparison purposes:

- **Learning-free methods.** We considered Kalman filter, constant velocity [18] and ORCA [19].
- **Vanilla LSTM.** An architecture with a LSTM encoder, a LSTM decoder, and no interaction module (each pedestrian is considered independently).
- **AMENet [28]**, a conditional variational auto-encoder based on attentive dynamic maps for interaction modeling, **AIN [29]**, an encoder-decoder pipeline focusing on global spatio-temporal interactions and **PecNet [30]**, a conditioned-on-goal endpoint variational auto-encoder. We reference the scores that are on the public leaderboard for AMENet and the ones referenced in [5] for AIN and PecNet.
- **Social NCE [17].** Best submission on the public leaderboard, with respect to FDE. It uses social pooling and contrastive learning. We reference the scores that are on the public leaderboard.

Table I shows the results on the four metrics and helps understand the pros and cons of each method. In terms of FDE, the Kalman filter is by far the worst of all, almost 30 cm behind constant velocity (but Type III scenes, on which evaluation is performed, are by definition scenes where trajectories cannot be correctly predicted using a Kalman filter). The constant velocity method is both extremely simple and reasonably effective, but at the cost of high collision rates. ORCA allows to completely get rid of collisions without sacrificing FDE. Vanilla LSTM is completely irrelevant, since it is worse even than the constant velocity method, highlighting how the potential of RNNs can only be revealed by using interaction encoders. Finally, the best submission on the leaderboard reaches a FDE that is 30 cm below the constant velocity method, with a Col-I of only 5%; however, as we said, ADE and FDE are still relatively high in absolute terms.

C. Experiments details

For training, we used ETH, UCY, WildTrack, L-CAS, and only part of CFF datasets, totalling ∼29000 scenes in the training set and ∼5000 scenes in the validation set. In the training procedure, we decrease the learning rate when the validation loss reaches a plateau, and also apply early-stopping when the validation loss stops decreasing for several epochs. We also use rotation augmentation as a data augmentation technique to regularize all the models.

We did not code everything from scratch, but rather built on top of the numerous baselines that are available with Trajnet++. Since out goal was not to beat the state-of-the-art but rather to allow meaningful comparison between different motion encoders, comparisons of given approaches are relevant given the same interaction module and hyper-parameter settings.

We tested the following architectures, denoted by their Encoder-Decoder structure. For each architecture, RNN can be replaced by either GRU [31], [32] or LSTM [33]:

- **RNN - RNN.** A common baseline.
- **Bi-RNN - RNN.** We used concatenation in order to fuse the outputs of the Bi-RNN, since it worked better than summation.
- **U-RNN - RNN.** The architecture described in Section III.

TABLE II

Comparison of motion-encoding designs with respect to various interactions modules architectures on interacting trajectories of TrajNet++ real world dataset.

Model **(Encoder - Decoder)**	**Interaction**	**ADE (m)** ± 0.01 m	**FDE (m)** ± 0.01 m	**Col-I (%)** ± 0.5%	**Col-II (%)** ± 1%
Constant velocity [18]	None	0.68	1.42	14.3	15.2
None - GRU	Dir. pooling [5]	0.63	1.33	6.9	12.1
LSTM - LSTM	Occ. pooling [3]	0.58	1.23	11.5	13.9
U-LSTM - LSTM	Occ. pooling [3]	**0.57**	**1.22**	**10.2**	14.9
GRU - GRU	Dir. pooling [5]	0.58	1.24	6.5	12.4
Bi-GRU - GRU	Dir. pooling [5]	0.59	1.26	6.7	11.7
U-GRU - GRU	Dir. pooling [5]	0.58	1.25	6.5	11.7
reversed U-GRU - GRU	Dir. pooling [5]	0.58	1.25	6.5	**11.0**
LSTM - LSTM	Dir. pooling [5]	0.58	1.25	6.4	11.4
Bi-LSTM - LSTM	Dir. pooling [5]	0.59	1.28	6.2	11.9
U-LSTM - LSTM	Dir. pooling [5]	**0.56**	**1.22**	**5.2**	11.9
reversed U-LSTM - LSTM	Dir. pooling [5]	0.58	1.26	6.6	**11.1**
LSTM - LSTM	Soc. pooling [3]	0.55	1.18	6.9	12.7
U-LSTM - LSTM	Soc. pooling [3]	**0.53**	**1.15**	**6.5**	11.5
Social NCE [17]	Soc. pool. [3] + contr. learning	**0.53**	**1.14**	**5.3**	11.3

- **reversed U-RNN - RNN.** The backward pass and forward pass are inverted in the U-RNN, in order to investigate if there is indeed a preferred direction of U-RNNs according to the data.

We used default number of parameters that were similar to the baselines in [5] and did not change between different models. However, this led to LSTM models having higher total number of parameters than their GRU counterparts, but it did not affect our conclusions. The order of magnitude of the uncertainties on the metrics were ± 1 cm on ADE and FDE, ± 0.5% on Col-I and ± 1% Col-II.

D. *Results*

In Table II, we present the results that we obtained during our experiments. The first thing to notice is that using a simple RNN decoder with *directional* pooling, even without an encoder, improved FDE by 10 cm and cuts Col-I by half compared to the Constant velocity model or to Vanilla LSTM. Secondly, adding a RNN encoder for past coordinates helped improving performance, which indicates that there is indeed relevant information in past positions. This suggests that pedestrians engage in complex trajectories that may span on relatively long durations.

Note that the proposed asymmetrical architecture is independent of the chosen recurrent unit. We observed in preliminary experiments that the encoder's architecture did not seem to have any impact, with identical performances of GRU - GRU, Bi-GRU - GRU, U-GRU - GRU and reversed U-GRU - GRU architectures. At first glance, one could conclude that the information contained in past coordinates may be too redundant to allow to detect any difference between encoder architectures, as there would be no further information to extract. Or that contrary to vehicles for example, pedestrian trajectories are too irregular to make good use of past information. However, experiments with LSTMs gave different results. LSTM - LSTM and Bi-LSTM - LSTM performed similarly as GRU architectures, but using a U-LSTM encoder helped get significantly better ADE, FDE and Col-I for *directional* pooling, suggesting that there was indeed unused information in past trajectories. Regarding Col-II, the best architectures seem to differ compared to the other metrics, but this appears to be non-significant given the small score differences and the order of magnitude of the standard deviations.

The better performance of U-LSTM compared to U-GRU strongly indicates that the additional information extracted by the U-RNN architecture came from long-term dependencies. Moreover, the hypothesis we proposed, that the non-symmetrical architecture of U-RNN should better leverage information by using the preferred direction of the data is supported by the absence of performance improvement when using a reversed U-LSTM encoder.

Since it was clear that, for the *directional* pooling case, the proposed Asymmetrical Bi-RNNs motion encoder performed better than regular LSTMs which are the *de facto* RNNs for trajectory encoding, we experimented U-LSTMs with *occupancy* and *social* pooling. In both experiments, our sequence encoder yielded significantly better results compared to regular LSTMs for every available metric (ADE,FDE, Col I). This suggests that the proposed architecture is a viable alternative to LSTMs for trajectory encoding.

V. Conclusion

In this article, we proposed a sequence encoder based on Asymmetrical Bi-RNNs to predict future pedestrians trajectories using naturalistic pedestrian scenes data from the widely studied Trajnet++ dataset. Contrary to many previous studies that proposed new interactions modules, our work solely relies on proposing a new sequence encoder that could easily be applied to all models that use the encoder-decoder pipeline for pedestrian trajectory forecasting, while taking advantage of the research on interactions and multi-modal trajectory prediction. The proposed sequence encoder was shown to achieve better prediction accuracy than previous sequence encoders such as LSTMs for a variety of existing approaches and interactions modules. This suggests that there is still room for improvement in coordinates-only approaches, and indicates that interactions are not the only aspect on which pedestrian trajectory prediction can progress. Although this work is highly preliminary, we believe our quantitative results could open many perspectives for future research. The success of Asymmetrical Bi-LSTMs compared to Asymmetrical Bi-GRUs suggests that this boost may come from using information with long-term dependencies, confirming that some pedestrians movements are influenced by long-term anticipation. We believe that these results constitute a promising baseline to replace LSTMs for a variety of approaches and could be used to significantly improve current pedestrian trajectory prediction algorithms.

References

[1] "Uber atg safety report 2020." [Online]. Available: https://uber.app.box.com/v/uberatgsafetyreport

[2] "Waymo safety report 2021." [Online]. Available: https://downloads.ctfassets.net/sv23gofxcuiz/4gZ7ZUxd4SRj1D1W6z3rpR/2ea16814cdb42f9e8eb34cae4f30b35d/2021-03-waymo-safety-report.pdf

[3] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social LSTM: Human Trajectory Prediction in Crowded Spaces," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 961–971, iSSN: 1063-6919.

[4] A. Gupta, J. Johnson, L. Fei-Fei, S. Savarese, and A. Alahi, "Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 2255–2264, iSSN: 2575-7075.

[5] P. Kothari, S. Kreiss, and A. Alahi, "Human trajectory forecasting in crowds: A deep learning perspective," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–15, 2021.

[6] F. Bartoli, G. Lisanti, L. Ballan, and A. Del Bimbo, "Context-aware trajectory prediction," in *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018, pp. 1941–1946.

[7] M. Pfeiffer, G. Paolo, H. Sommer, J. Nieto, R. Siegwart, and C. Cadena, "A data-driven model for interaction-aware pedestrian motion prediction in object cluttered environments," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5921–5928.

[8] A. Vemula, K. Muelling, and J. Oh, "Social attention: Modeling attention in human crowds," in *2018 IEEE international Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 4601–4607.

[9] Y. Ma, E. W. M. Lee, and R. K. K. Yuen, "An artificial intelligence-based approach for simulating pedestrian movement," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3159–3170, 2016.

[10] [Online]. Available: https://www.kaggle.com/c/riiid-test-answer-prediction/discussion/209581

[11] M. Schuster and K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, no. 11, pp. 2673–2681, 1997.

[12] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[13] H. Xue, D. Q. Huynh, and M. Reynolds, "Bi-prediction: pedestrian trajectory prediction based on bidirectional lstm classification," in *2017 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*. IEEE, 2017, pp. 1–8.

[14] Y. Yao, E. Atkins, M. Johnson-Roberson, R. Vasudevan, and X. Du, "Bitrap: Bi-directional pedestrian trajectory prediction with multi-modal goal estimation," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1463–1470, 2021.

[15] S. Becker, R. Hug, W. Hübner, and M. Arens, "RED: A Simple but Effective Baseline Predictor for the TrajNet Benchmark," in *Computer Vision – ECCV 2018 Workshops*, L. Leal-Taixé and S. Roth, Eds. Cham: Springer International Publishing, 2019, vol. 11131, pp. 138–153, series Title: Lecture Notes in Computer Science.

[16] F. Giuliari, I. Hasan, M. Cristani, and F. Galasso, "Transformer networks for trajectory forecasting," in *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE, 2021, pp. 10 335–10 342.

[17] Y. Liu, Q. Yan, and A. Alahi, "Social NCE: Contrastive Learning of Socially-aware Motion Representations," *arXiv:2012.11717 [cs]*, Dec. 2020.

[18] C. Schöller, V. Aravantinos, F. Lay, and A. Knoll, "What the constant velocity model can teach us about pedestrian motion prediction," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1696–1703, 2020.

[19] J. van den Berg, S. J. Guy, M. Lin, and D. Manocha, "Reciprocal n-body collision avoidance," in *Robotics Research*, C. Pradalier, R. Siegwart, and G. Hirzinger, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 3–19.

[20] N. Nikhil and B. Tran Morris, "Convolutional neural network for trajectory prediction," in *Proceedings of the European Conference on Computer Vision (ECCV) Workshops*, September 2018.

[21] T. Fernando, S. Denman, S. Sridharan, and C. Fookes, "Neighbourhood context embeddings in deep inverse reinforcement learning for predicting pedestrian motion over long time horizons," in *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 2019, pp. 1179–1187.

[22] A. Rudenko, L. Palmieri, M. Herman, K. M. Kitani, D. M. Gavrila, and K. O. Arras, "Human motion trajectory prediction: a survey," *The International Journal of Robotics Research*, vol. 39, no. 8, pp. 895–935, Jul. 2020, publisher: SAGE Publications Ltd STM.

[23] S. Pellegrini, A. Ess, and L. Van Gool, "Improving data association by joint modeling of pedestrian trajectories and groupings," in *European conference on computer vision*. Springer, 2010, pp. 452–465.

[24] L. Leal-Taixé, M. Fenzi, A. Kuznetsova, B. Rosenhahn, and S. Savarese, "Learning an image-based motion context for multiple people tracking," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3542–3549.

[25] T. Chavdarova, P. Baqué, S. Bouquet, A. Maksai, C. Jose, T. Bagautdinov, L. Lettry, P. Fua, L. Van Gool, and F. Fleuret, "Wildtrack: A multi-camera hd dataset for dense unscripted pedestrian detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5030–5039.

[26] 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 *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 5942–5948.

[27] A. Alahi, V. Ramanathan, and L. Fei-Fei, "Socially-aware large-scale crowd forecasting," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 2203–2210.

[28] H. Cheng, W. Liao, M. Y. Yang, B. Rosenhahn, and M. Sester, "Amenet: Attentive maps encoder network for trajectory prediction," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 172, pp. 253–266, 2021.

[29] Y. Zhu, D. Ren, M. Fan, D. Qian, X. Li, and H. Xia, "Robust trajectory forecasting for multiple intelligent agents in dynamic scene," *arXiv preprint arXiv:2005.13133*, 2020.

[30] K. Mangalam, H. Girase, S. Agarwal, K.-H. Lee, E. Adeli, J. Malik, and A. Gaidon, "It is not the journey but the destination: Endpoint conditioned trajectory prediction," in *European Conference on Computer Vision*. Springer, 2020, pp. 759–776.

[31] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[32] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

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