Time2Vec Embedding on a Seq2Seq Bi-directional LSTM Network for Pedestrian Trajectory Prediction

Victor Peñaloza[0000−0001−7335−8255]

RLICT: Research Laboratory in Information and Communication Technologies.
Unversidad Galileo, Ciudad de Guatemala, Guatemala.
victorsergio@galileo.edu

Abstract. This paper presents a description of a proposed architecture for the pedestrian trajectory prediction task on the TrajNet dataset. This task aims to determine future final agent position and future agent movement trajectory using previous agent positional information. The proposed architecture is a Seq2Seq bi-directional LSTM (Long short-term memory) network with a Time2Vec input embedding layer. After experimenting and evaluating, our experimental results show that adding a Time2Vec input-embedding layer improves the performance of a Seq2Seq bi-directional LSTM recurrent model on the pedestrian trajectory prediction task.

Keywords: Pedestrian · Trajectory Prediction · Time2Vec · Embedding · Sequence to Sequence · TrajNet.

1 Introduction

Many researchers have studied and discussed human mobility analysis due to its applications in robotics, autonomous driving systems, and mobile telecommunications systems. With the growing size of collected data captured by sensors and smartphones at present, it is possible to gather much data about human mobility to be processed later to identify patterns in data [30]. However, some applications need to have a system capable of anticipating and determining a future human position in an online way, such as the case of autonomous driving systems, in which it is helpful to know in advance the pedestrian movements in streets to avoid accidents.

In this paper, we focused on predicting the future path for an agent, using previous movement information to anticipate the future movement. The contributions of this work can be summarized as follows:

1. Show a simple Seq2Seq architecture to predict future pedestrian movement, using an encoder-decoder bi-directional LSTM (Long short-term memory) recurrent model with an input embedding layer.

2. Compare the performance between a model using an input embedding layer and a model without it. We show experimentally that for pedestrian path prediction on TrajNet [31] dataset, the use of Time2Vec [20] embedding layer improves the Seq2Seq model results.

This paper comprises eight sections: the first one presents an introduction to this task and study. The second section describes some previous work in the pedestrian trajectory prediction task. The third section describes the problem statement, metrics, and used dataset. The fourth section presents some informational background of blocks used to build the proposed system. The fifth section describes the system architecture proposed. The sixth section describes the experiments performed. The seventh section shows the results achieved in the model comparisons. The last section presents some conclusions and future work to continue the experiments on this task.

2 Related Work

Many deep learning models have been proposed for human trajectory prediction with data-driven approaches to learning the human motion. LSTM [18] based architectures have been shown to be capable of learning general human movement and can predict their future trajectories [2]. Simple encoder-decoder architectures have been shown promising results in pedestrian trajectory prediction, using only position information about pedestrians [6]. Besides, more complex approaches using Generative Adversarial Networks [15] have been used to predict the motion of pedestrians interacting with others [4].

3 Problem Statement

3.1 Problem Formulation

The goal of this task is to predict a future agent trajectory without using human-human interaction data. We formulate this problem as a sequence generation problem, where we assume agent input trajectories as a sequence $X = (X_1, X_2, \ldots, X_8)$ to predict a future trajectory $X_{pred} = (X_9, X_{10}, \ldots, X_{20})$. Whereas $X_i = (x_i, y_i)$ are (x, y) coordinates in a 2D plane for any time-instant i.

3.2 Dataset Description

For the experiments, we used the TrajNet Benchmark Dataset [31]. This dataset bundle the most common datasets used for pedestrian trajectory analysis. TrajNet challenge also provides a common platform to compare path prediction approaches. We train our models using the World H-H TRAJ dataset, in which the pedestrian trajectories are formed by x, y world coordinates in meters.

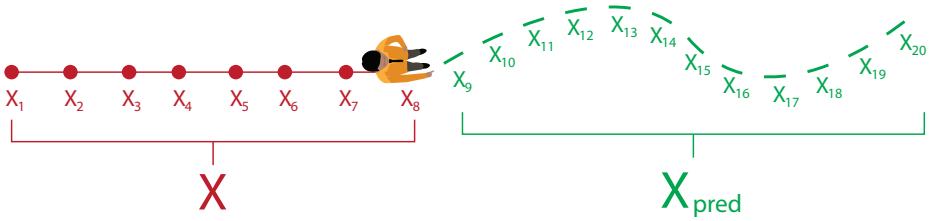


Fig. 1: This task's main goal is to use eight previous steps to predict a future agent trajectory comprised of 12 steps. The agent position on each step is defined by $X_i = (x_i, y_i)$ a coordinate in a 2D plane.

3.3 Metrics and Evaluation

We use the following error metrics to evaluate the models' performance over a test dataset: average displacement error (ADE) and the final displacement error (FDE). ADE is defined as the average of Euclidian distances between ground truth and the prediction overall predicted time steps. FDE is defined as the Euclidian distance between the predicted final position and the ground truth final position. We observed eight sequential picture frames (*2.8 seconds*) to form an input sequence to predict a future sequence comprised of 12 sequential picture frames (*4.8 seconds*).

4 Background

To better understand, this section introduces some fundamental concepts of the blocks used to build our model. The following concepts are presented: LSTM [18], bi-directional LSTM [33], Seq2Seq learning, embedding, and Time2Vec [20].

4.1 LSTM, Long Short-Term Memory

Neural Recurrent Networks (RNN) has been proved to be an effective architecture for diverse learning problems with sequential data [19]. LSTM networks were designed to deal with vanishing gradients problem in learning long-term dependencies [18]. However, LSTMs can still suffer from instability with improper initialization [27]. Additionally, LSTM architectures are highly complex, having about four times more parameters than a simple RNN with an equal number of hidden layers [29]; therefore, LSTM architectures have higher memory requirements [32].

LSTM architecture is mainly comprised of a memory cell and unit gates. The network can use these gates to decide when to keep or overwrite information in the memory cell and decide when to prevent perturbing other cells by a memory cell output [18].

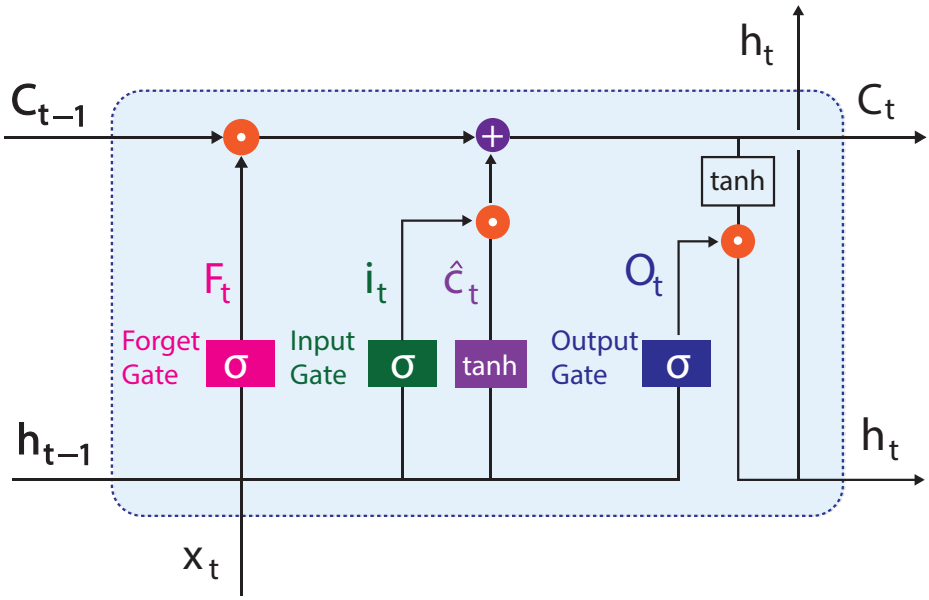


Fig. 2: The architecture of a LSTM block.

4.2 BLSTM, bi-directional LSTM

A bi-directional RNN(BRNN) [33] has the complete information from the past and the future for every point in a given sequence. A bi-directional RNN(BRNN) is comprised of two separate recurrent neural nets, both of which are connected to the same output layer. Each training sequence forwards and backward feeds the bi-directional RNN (BRNN). Thus for every point in a given sequence, a bi-directional RNN(BRNN) has complete, sequential information about all points before and after it [16]. One limitation of the bi-directional RNN(BRNN) is that it is not appropriate for the online setting, as it is implausible to know sequence elements that have not been observed [25]. Nevertheless, we must distinguish between tasks that require an output after every input (true online tasks) and those where outputs are only needed at the end of some input segment [16]. This paper's pedestrian trajectory task is in the second class; therefore, the bi-directional RNN(BRNN) can be used without a problem. The LSTM and BRNN are compatible, and the conjunction of these two ideas is termed as BLSTM (bi-directional LSTM) [16].

4.3 Recurrent Sequence-to-Sequence Learning

Seq2Seq is a structure designed to modeling sequential problems in which input sequences and outputs sequences are variable in length [36].

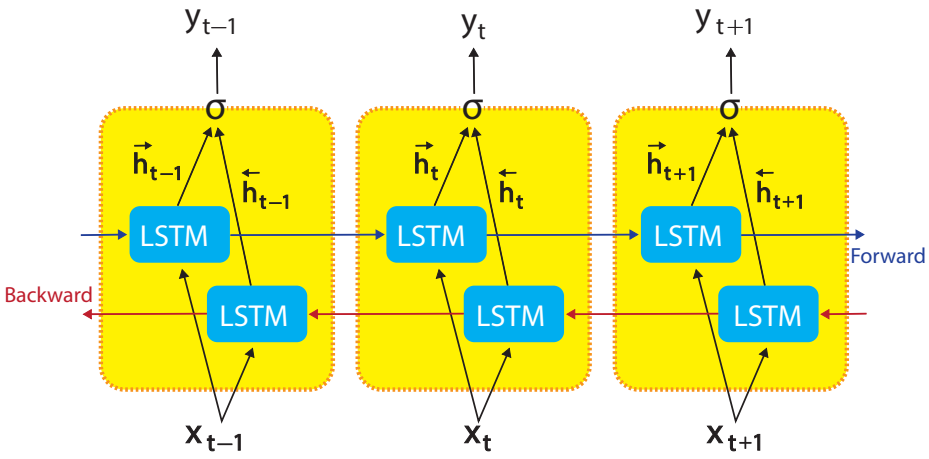


Fig. 3: Bi-directional LSTM architecture.

The most common structure associated with Seq2Seq learning is an encoder-decoder structure with recurrent neural networks [34, 5]. Being an LSTM [18] the most used neural recurrent unit in the encoder and decoder section.

The encoder processes an input sequence of m elements and returns representations of z in a fixed-length vector. The decoder takes z and generates an output sequence element at a time [13].

Despite the success of this type of architecture in machine translation tasks, due to the fact of compressing all the necessary information of a source sentence into a fixed-length vector, it could be hard for the neural network to cope with long sentences, deteriorating the performance of the model as the length of an input sentence increases [5, 8]. Different attention-based models were successfully applied to overcome this [5, 26]. Attention-based models are an extension for the encoder-decoder architecture that allows the model to automatically search for relevant parts of a source sentence to predict a target element without encoding a whole input sentence into a single fixed-length vector [5].

4.4 Embedding

Many factors affect the Machine Learning model's performance; among these factors, the principal factor is the representation and quality of the data used for training [21].

For example, in the Natural Language Processing (NLP) area, sequences of sentences, words, or characters are used as input to recurrent neural networks based models; generally is desired to provide a compact representation of these features instead of sparse representation [35]. Diverse techniques have been proposed for feature representation in NLP as one-hot encoding representation [38], continuous bag of words(CBOW) [28], and word-level embedding [14].

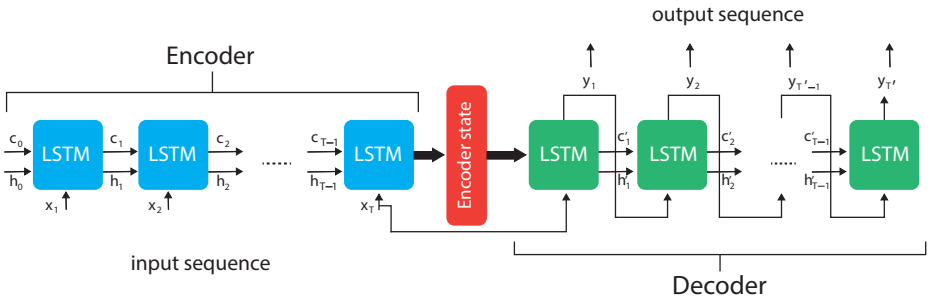


Fig. 4: Encoder-Decoder structure from a Seq2Seq model.

Word level embedding goal is to learn a representation in which ideally words semantically related are highly correlated in the representation space [35]; in this way, good results have been obtained in classification [37, 24, 12] and sentiment analysis NLP tasks [10, 3].

4.5 Time2Vec Embedding

In many applications with sequential data, time is an important feature. In many models using recurrent neural networks, the time is not taken in count as a feature. When time is observed as an essential factor, this is generally added as an additional feature [9, 11, 23].

Time2Vec is a vector representation (*embedding*) to time that it can be combined easily with other architectures [20].

Time2Vec as a time representation was designed with three essential properties in mind: 1- periodicity, the capability to capture periodic and non-periodic patterns; some events can occur periodically(e.g., the weather over different seasons), but others events may be non-periodic and only happens after a point in time, 2- being invariant to time rescaling, since time can be measured in different scales(e.g., days, hours, seconds), and 3- simplicity, should be easy to be added to existent architectures [20].

For a given scalar notion of time τ, Time2Vec of τ, denoted as $\mathbf{t2v}(\tau)$, is a vector of size $k+1$, defined as follows:

$$\mathbf{t2v}(\tau)[i] \begin{cases} \omega_i \tau + \varphi_i, & \text{if } i = 0 \\ \mathcal{F}(\omega_i \tau + \varphi_i), & \text{if } 1 \leq i \leq k. \end{cases}$$

Where $\mathbf{t2v}(\tau)[i]$ is the i^{th} element of $\mathbf{t2v}(\tau)$, $\mathcal{F}$ is a periodic activation function, and ω_is and φ_is are learnable parameters.

When using $\mathcal{F} = sine$, for $1 \leq i \leq k$, ω_i and φ_i are the frequency and the phase-shift of the sine function.

This function permits learn periodic behaviors. The linear term section can be used to represent no periodic behaviors [20].

5 System description

The main idea of the proposed system is the combination of already existing techniques that have shown improvements in other research areas as Natural Language Processing (NLP), and apply these techniques to the pedestrian trajectory prediction task.

We proposed the use of a recurrent Seq2Seq architecture that uses bi-directional LSTM [33, 18] layers to build the encoder and decoder with a Time2Vec embedding layer on encoder inputs to feed a better positional sequence representation to the model. Fig. 5 shows the proposed architecture for pedestrian trajectory prediction.

We choose a Seq2Seq multi-step architecture because it has been observed that single-step prediction models that use windowing to predict a complete future path are prone to error accumulation, and Seq2Seq models have been showing better results in similar tasks [36].

Although Time2Vec originally was proposed for a better time representation, we can see the sequences of coordinates (*positions in the space*) as signals that include an implicit time relation and use Time2Vec embedding to create an embedding that represents this signal with a learnable frequency and phase shifts of *sine* function.

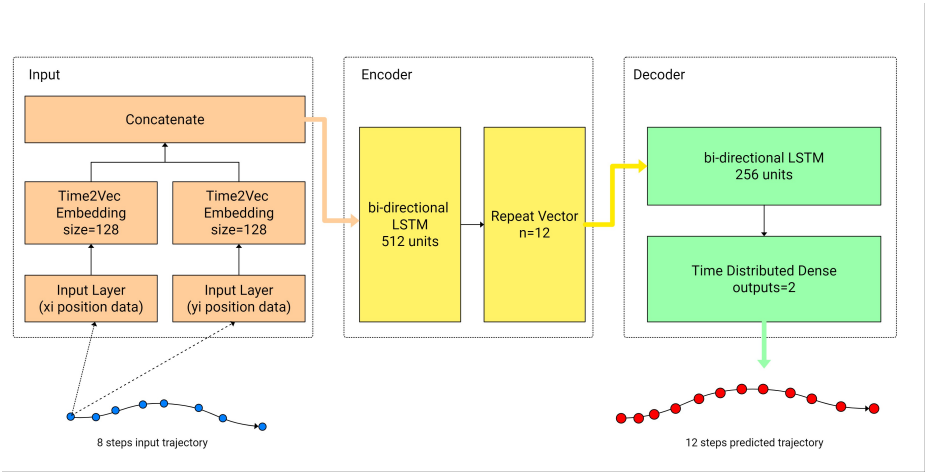


Fig. 5: Proposed Seq2Seq + Time2Vec embedding architecture.

6 Experiments

For performance comparison purposes, we build a Seq2Seq bi-directional LSTM architecture that not uses Time2Vec embedding and compares it against the

proposed Seq2Seq bi-directional LSTM model with Time2Vec embedding to observe if the use of a Time2Vec embedding is beneficial to the performance of the Seq2Seq model in trajectory prediction results.

For a better comparison, all models were trained for 81 epochs (*one epoch is a full training pass over the entire dataset such that each example has been seen once*).

The models were trained using a learning rate equal to 1e-5, a batch size equal to 8, and MSE as a loss function.

For comparison, we used the error metrics: average displacement error (ADE) and the final displacement error (FDE) in meters.

We run various experiments using different embedding vector sizes, 128, 256, and 512, respectively, to compare the effect of embedding size in the model's performance.

All the models were created using TensorFlow [1], running on Google Colab Environment, and using Weight and Biases [7] for monitoring training metrics.

7 Results

Table 1: Comparison of prediction error among different models to evaluate embedding performance.

Model	Embedding size	ADE	FDE
RED Predictor	-	***0.364***	***1.229***
Social Forces (EWAP)	-	0.371	1.266
Social LSTMv2	-	0.675	2.098
Seq2Seq + Time2Vec	512	**0.551**	**1.637**
Seq2Seq + Time2Vec	256	0.611	1.67
Seq2Seq + Time2Vec	128	0.658	1.83
Seq2Seq	-	0.767	2.202

Table 1 shows our experimental results; it can be observed that using Time2Vec embedding improves the Seq2Seq model in predicting more accurate trajectories. Time2Vec embedding helps the model feeding the Seq2Seq model with a better representation of positional trajectories sequences.

Notice that runs with bigger embedding vector sizes improves both final displacement error (FDE) and average displacement error (ADE) results.

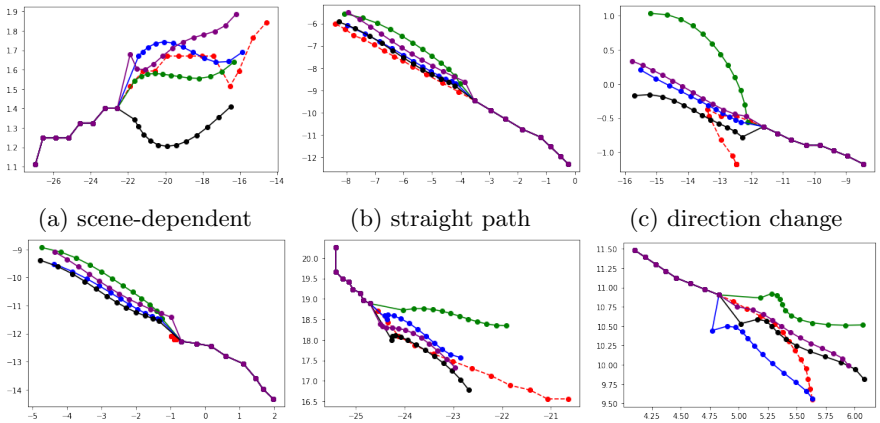
All results from Table 1 were calculated using a test set provided by the TrajNet Official Benchmark website [31].

Additionally, the Table 1 shows results achieved by state of the art methods for comparison. We compare our model with the best models with references found on Trajnet Benchmark Dataset [31] for a fair comparison.

However, this comparison must be taken carefully since other methods as Social LSTM [2] and Social Forces (EWAP) [17] use pedestrian interaction in-

formation, while our proposed method only uses pedestrian positional information. The closest and fair comparison is against RNN-Encoder-Dense (RED) predictor [6] due to its simplicity and similarity, using a recurrent-encoder with a dense multi-layer perceptron (MLP) layer stacked architecture and using only positional information.

Notice that, although our results are lower than the state of the art models in the pedestrian trajectory prediction task, this work aimed to research better representations for positional data in pedestrian trajectories.



(a) scene-dependent (b) straight path (c) direction change

(d) sudden stop (e) speed change (f) final position

Fig. 6: Some examples of hard to predict trajectories for the models. The ground truth trajectories are shown in red, and the model's predictions are: 128 embedding dimension (blue), 256 embedding dimension (black), 512 embedding dimension (purple), and no-embedding (green). The scale of the graph axis is in meters.

Fig. 6 shows trajectories hard to predict for the models. In Fig. 6c, it can be observed a sudden change of direction; the pedestrian decides to return on his path; this type of behavior is hard to predict even for humans. Fig. 6d depicts another hard to predict case; in this scenario, the pedestrian stop could be caused for a red traffic light. To predict this trajectory correctly, additional information for the model as traffic light state information could be beneficial. For the case shown in Fig. 6e, it can be observed that the pedestrian decides to increment its speed suddenly; for example due to the pedestrian was in a hurry, it is hard to predict this behavior due to the prediction was made using the previous constant speed. This problem could be addressed by creating models that could predict trajectories using small future step windows. It should be noticed that it is easy for the models to predict the movement when the speed is constant and follows a commonly used trajectory in the scene; for example, as shown in Fig. 6b the

pedestrian follows a sidewalk and maintains a constant speed. In these types of cases, all the models exhibit similar results in the predictions.

It is worth to notice that the models learn scene dependent features, as shown in Fig. 6a, and different models could predict different paths due to obstacles found in the scene. It could be worthy of continuing research this problem with multi-modal models that could output a probability distribution over different possible paths. At last, in some cases, the model gives more importance to predict a correct final position instead of predicting intermediate steps correctly, as shown in Fig. 6f.

Table 2: Average displacement error (ADE) and final displacement error (FDE) for the special cases prediction shown in Fig. 6

embedding size	scene dependent		straight path		direction change		sudden stop		speed change		final position	
	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE	ADE	FDE
none	0.625	1.739	0.517	0.563	1.003	3.497	1.618	4.912	0.786	2.190	0.306	1.054
128	**0.533**	**1.285**	0.503	0.444	**0.785**	**3.346**	**1.292**	**4.271**	0.799	2.44	0.138	**0.010**
256	0.759	1.956	**0.327**	**0.157**	0.815	3.382	1.469	4.685	**0.617**	**2.049**	**0.120**	0.516
512	0.577	1.639	0.381	0.682	0.941	3.638	1.502	4.529	0.78	2.449	0.126	0.537

In Table 2 are the average displacement error (ADE) and final displacement error(FDE) for the mentioned special cases. Note that in these special cases, the use of bigger embedding does not mean better prediction results. However, as observed in Table 1, on average, evaluating over the complete test set, the use of bigger embedding achieves better prediction results. This contradiction makes us think that the dataset comprises a large variety of cases, in which some of them are easy to predict for some models, and others are hard for them. A possible solution could be to use an ensemble of models to select the best prediction among different models that could be successful in specialized cases.

8 Conclusions and Future Work

This work described an architecture that uses a Seq2Seq bi-directional LSTM recurrent model with an input embedding layer to be compared with a similar architecture without an input embedding layer to show the benefits of having an adequate input data representation for models on the pedestrian trajectory prediction task.

According to our experimental results, although embedding improves the proposed architecture's performance in this task, now the embedding vector size hyperparameter appears. It can be observed that the size of this vector has a model's performance impact and must be adjusted correctly.

Additionally, it can be worth testing other architectures as Generative Adversarial Network (GAN) [15] based, transformed based, and Convolutional Neural

Network (CNN) [22] based on comparing embedding performance in the pedestrian trajectory prediction task.

Acknowledgments

This work was supported by Facultad de Ingeniería de Sistemas, Informática y Ciencias de la Computación (FISICC) and Research Laboratory in Information and Communication Technologies (RLICT), both part of Universidad Galileo from Guatemala.

We thank Jean-Bernard Hayet, PhD. for helpful advice and research directions related to this work.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), https://www.tensorflow.org/, software available from tensorflow.org
2. Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., Savarese, S.: Social lstm: Human trajectory prediction in crowded spaces. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 961–971 (2016)
3. Altowayan, A.A., Tao, L.: Word embeddings for arabic sentiment analysis. 2016 IEEE International Conference on Big Data (Big Data) pp. 3820–3825 (2016)
4. Amirian, J., Hayet, J.B., Pettré, J.: Social ways: Learning multi-modal distributions of pedestrian trajectories with gans. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW) pp. 2964–2972 (2019)
5. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. CoRR **abs/1409.0473** (2015)
6. Becker, S., Hug, R., Hübner, W., Arens, M.: Red: A simple but effective baseline predictor for the trajnet benchmark. In: ECCV Workshops (2018)
7. Biewald, L.: Experiment tracking with weights and biases (2020), https://www.wandb.com/, software available from wandb.com
8. Cho, K., Merrienboer, B.V., Bahdanau, D., Bengio, Y.: On the properties of neural machine translation: Encoder-decoder approaches. In: SSST@EMNLP (2014)
9. Choi, E., Bahadori, M.T., Schuetz, A., Stewart, W.F., Sun, J.: Doctor ai: Predicting clinical events via recurrent neural networks. Proceedings of machine learning research **56**, 301–318 (2016), https://app.dimensions.ai/details/publication/pub.1084501054
10. Deho, B.O., Agangiba, A.W., Aryeh, L.F., Ansah, A.J.: Sentiment analysis with word embedding. 2018 IEEE 7th International Conference on Adaptive Science & Technology (ICAST) pp. 1–4 (2018)
11. Du, N., Dai, H., Trivedi, R., Upadhyay, U., Gomez-Rodriguez, M., Song, L.: Recurrent marked temporal point processes: Embedding event history to vector. In:

Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 1555–1564. KDD '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2939672.2939875, https://doi.org/10.1145/2939672.2939875
12. Ge, L., Moh, T.: Improving text classification with word embedding. In: 2017 IEEE International Conference on Big Data (Big Data). pp. 1796–1805 (2017)
13. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.: Convolutional sequence to sequence learning. In: ICML (2017)
14. Goldberg, Y.: Neural network methods for natural language processing. Synthesis Lectures on Human Language Technologies **10**(1), 1–309 (2017). https://doi.org/10.2200/S00762ED1V01Y201703HLT037, https://doi.org/10.2200/S00762ED1V01Y201703HLT037
15. Goodfellow, I.J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A.C., Bengio, Y.: Generative adversarial nets. In: NIPS (2014)
16. Graves, A., Schmidhuber, J.: Framewise phoneme classification with bidirectional lstm and other neural network architectures. Neural Networks **18**(5), 602 – 610 (2005). https://doi.org/https://doi.org/10.1016/j.neunet.2005.06.042, http://www.sciencedirect.com/science/article/pii/S0893608005001206, iJCNN 2005
17. Helbing, D., Molnár, P.: Social force model for pedestrian dynamics. Phys. Rev. E **51**, 4282–4286 (May 1995). https://doi.org/10.1103/PhysRevE.51.4282, https://link.aps.org/doi/10.1103/PhysRevE.51.4282
18. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**, 1735–1780 (1997)
19. Karpathy, A., Johnson, J.E., Fei-Fei, L.: Visualizing and understanding recurrent networks. ArXiv **abs/1506.02078** (2015)
20. Kazemi, S., Goel, R., Eghbali, S., Ramanan, J., Sahota, J., Thakur, S., Wu, S., Smyth, C., Poupart, P., Brubaker, M.A.: Time2vec: Learning a vector representation of time. ArXiv **abs/1907.05321** (2019)
21. Kotsiantis, S.B., Kanellopoulos, D., Pintelas, P.E.: Data preprocessing for supervised leaning. World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering **1**, 4104–4109 (2007)
22. LeCun, Y., Haffner, P., Bottou, L., Bengio, Y.: Object recognition with gradient-based learning. In: Shape, Contour and Grouping in Computer Vision (1999)
23. Li, Y., Du, N., Bengio, S.: Time-dependent representation for neural event sequence prediction (2018), https://openreview.net/pdf?id=HyrT5Hkvf
24. Lilleberg, J., Zhu, Y., Zhang, Y.: Support vector machines and word2vec for text classification with semantic features. In: 2015 IEEE 14th International Conference on Cognitive Informatics Cognitive Computing (ICCI*CC). pp. 136–140 (2015)
25. Lipton, Z.C.: A critical review of recurrent neural networks for sequence learning. ArXiv **abs/1506.00019** (2015)
26. Luong, T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. ArXiv **abs/1508.04025** (2015)
27. Mehdipour-Ghazi, M., Nielsen, M., Pai, A., Modat, M., Cardoso, M., Ourselin, S., Sørensen, L.: On the initialization of long short-term memory networks. ArXiv **abs/1912.10454** (2019)
28. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space (2013)
29. Mikolov, T., Joulin, A., Chopra, S., Mathieu, M., Ranzato, M.: Learning longer memory in recurrent neural networks. CoRR **abs/1412.7753** (2015)

30. Morzy, M.: Mining frequent trajectories of moving objects for location prediction. In: Perner, P. (ed.) Machine Learning and Data Mining in Pattern Recognition. pp. 667–680. Springer Berlin Heidelberg, Berlin, Heidelberg (2007)
31. Sadeghian, A., Kosaraju, V., Gupta, A., Savarese, S., Alahi, A.: Trajnet: Towards a benchmark for human trajectory prediction. arXiv preprint (2018)
32. Salehinejad, H., Baarbe, J., Sankar, S., Barfett, J., Colak, E., Valaee, S.: Recent advances in recurrent neural networks. ArXiv **abs/1801.01078** (2018)
33. Schuster, M., Paliwal, K.K.: Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing **45**(11), 2673–2681 (1997)
34. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. ArXiv **abs/1409.3215** (2014)
35. Torfi, A., Shirvani, R.A., Keneshloo, Y., Tavaf, N., Fox, E.A.: Natural language processing advancements by deep learning: A survey (2020)
36. Wang, C., Ma, L., Li, R., Durrani, T.S., Zhang, H.: Exploring trajectory prediction through machine learning methods. IEEE Access **7**, 101441–101452 (2019)
37. Şeyhmus Yilmaz, Toklu, S.: A deep learning analysis on question classification task using word2vec representations. Neural Computing and Applications **32**, 2909–2928 (2020)
38. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. In: Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1. p. 649–657. NIPS'15, MIT Press, Cambridge, MA, USA (2015)