Context-based cyclist path prediction using Recurrent Neural Networks

Ewoud A. I. Pool [1], Julian F. P. Kooij [1] and Dariu M. Gavrila [1]

***Abstract*— This paper proposes a Recurrent Neural Network (RNN) for cyclist path prediction to learn the effect of contextual cues on the behavior directly in an end-to-end approach, removing the need for any annotations. The proposed RNN incorporates three distinct contextual cues: one related to actions of the cyclist, one related to the location of the cyclist on the road, and one related to the interaction between the cyclist and the egovehicle. The RNN predicts a Gaussian distribution over the future position of the cyclist one second into the future with a higher accuracy, compared to a current state-of-the-art model that is based on dynamic mode annotations, where our model attains an average prediction error of 33 cm one second into the future.**

I. Introduction

An important case for self-driving vehicles is that they can reduce the number of traffic accidents. To facilitate a driving style that is not only safe but also comfortable and time-efficient, self-driving vehicles need to anticipate, i.e. predict where other road users are likely to be in the near future. Contextual information, such as spatial layout [1], [2], [3], or class-specific visual cues (a car blinker, an cyclists' outstretched arm, etc) [4] can be used to improve the accuracy of such predictions compared to using positional information only.

Vulnerable Road Users (VRUs) (i.e. pedestrians and cyclists) are particularly challenging road users to deal with. They can rapidly switch between various motion modes, such as walking or standing. Separate models are often learned for each of the identified modes [2], [5]. The various contextual cues need to be integrated into a predictive model to assess which mode the VRU will switch to, and when the VRU will do so, e.g. in a Dynamic Bayesian Network (DBN) [4]. Because of the small amount of parameters in these handcrafted models, their effectiveness can be shown even in small datasets crafted towards a specific scenario. They furthermore have the advantage that their inner working can be inspected, and their performance be explained (a plus in the context of functional safety).

On the other hand, handcrafted models require additional effort of identifying the different dynamic modes, as well as manually labeling the specific dynamic mode of each track at every moment. This scales poorly with dataset size. Furthermore, the switch between two modes is usually gradual, creating a gray area between two modes where it is not directly clear to which it should belong. As a result, the learned models for each mode will reflect the manual annotations rather than the real world situation. A solution for these issues is to use Recurrent Neural Networks (RNNs), which learn everything jointly in an end-to-end fashion. However, it is not known if an RNN can combine the various types of context cues in VRU path prediction on the small-scale datasets found for such context cues.

In this paper, we investigate if RNNs can incorporate context cues as effectively as the DBN from [4]. We show that the highest accuracy is obtained when all context cues are incorporated in the model, and outperform the DBN even on the small dataset. Additionally, we demonstrate that training without the annotated dynamic modes can be beneficial, revealing the actual moment where the dynamic modes switch. Finally, we explore the possible improvement in accuracy by increasing the amount of training data, and show that some contextual cues are more robust to anomalous behavior than others.

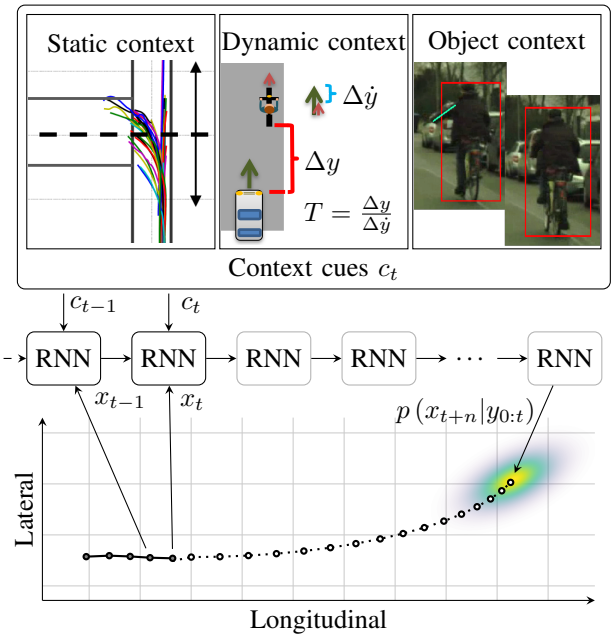


Fig. 1: A schematic overview of the position and context features used in the proposed RNN. In the used dataset, the static context is the distance to the intersection where the cyclist might turn left. The dynamic context is the time it would take for the vehicle to overtake the cyclist assuming they maintain their current velocity. The object context indicates the confidence of a trained detector whether the cyclist is raising their arm. The RNN receives this information at every time step, and combines it with the position of the cyclist to predict a Gaussian distribution over the future position of the cyclist.

1) Intelligent Vehicles group, Cognitive Robotics department, Technical University Delft, The Netherlands; {E.A.I.Pool, J.F.P.Kooij, D.M.Gavrila}@tudelft.nl;

II. Previous Work

In VRU path prediction, the goal is to model the dynamic behavior of VRUs in order to accurately predict their future location for time horizons longer than the immediate future (say, 0.5 s). The predictions can be improved by integrating various kinds of contextual cues. These are roughly divided into environment context cues and object-related context cues.

Environment context cues refers to the influence of the world surrounding the VRU on their path. These are static effects such as an expectation where VRUs prefer to walk to [3], or the VRU's preference to traverse certain kinds of semantic areas (sidewalks, grass, zebra crossings, etc.). One way of implementing this is implemented through Inverse Reinforcement Learning (IRL) [1], [6], but it is also possible to learn the weights directly using a neural network [7]. IRL is conditioned on the goal destination of the VRU, but it is possible to estimate the goal online jointly with the future path. For instance, Rehder *et al*. [8] estimate the goal using a neural net that combines image data with the previous path of the VRU. Ballan *et al*. [9] learn preferred routes directly on image data rather than the semantic information, and show that the learned knowledge is transferable to new locations. Another approach is to directly encode the structure of the road ahead to limit the possible paths that the VRU can take [2]. Dynamic objects can also influence the future path of VRUs. Social Force Models [10], [11] model the influence that nearby VRUs have on each other.

Object-related context cues relate to the visual appearance of the VRU himself. Many papers have focused on predicting whether a pedestrian will cross the street or not based on such features. Keller and Gavrila [12] do this by predicting the dense optical flow of the VRU's image, using Gaussian Process Dynamic Models (GPDM). Quintero *et al*. [5] use GPDM on a 3D joint model that is extracted from the visual and depth image of the pedestrian. VRU-related context cues also include whether the VRU is aware of his or her surroundings. Kooij *et al*. [4] incorporate both the risk of collision and the pedestrian's awareness thereof into a DBN to accurately predict the future position of a pedestrian who might cross the road. Additionally, they show the same DBN structure can also be used to predict the future position of a cyclist who might turn left at a coming intersection. A closely related field that also uses context cues is intent prediction for VRUs. Here, context cues can also be used to predict the intent of a VRU, such as the pose [13], or do the intention prediction directly on the image data [14].

Path prediction also requires a dynamic model to propagate the positional and contextual information into the future. One approach is to create separate models for different kinds of behavior and predict which behavior is currently most likely [4], [5], which makes the entire model more interpretable to users. However, due to the non-linearity in VRU path prediction some research has pursued RNNs to get an improved prediction accuracy, such as Long Short Term Memory networks (LSTMs) [10], [11]. Bhattacharyya *et al*. [15] use LSTMs to predict future positions of pedestrians directly in the image plane. Besides LSTMs, there are also other RNNs to process time series, such as a Gated Recurrent Unit (GRU) which uses fewer parameters while keeping a similar performance as an LSTM [16]. An RNN can also predict a Gaussian distribution directly [17]. RNNs cannot implicitly handle missing data. Posner and Ondrúška [18] add an extra binary input to each measurement to mark if the measurement has data or not. Fraccaro *et al*. [19] attempt to combine the best of Kalman filters and neural networks by assuming that the dynamic latent space of the neural net is in fact a Kalman filter, allowing them to use the exact inference, prediction and smoothing of a Kalman filter for the dynamics.

To train and validate the methods mentioned above, there are several datasets available, such as the Stanford Drone (SD) dataset [9], the Tsinghua-Daimler Cyclist (TDC) dataset [20] and the very recently released Eurocity Persons (ECP) dataset [21]. The SD dataset involves an aerial view from a high altitude however, and therefore does not contain any VRU-related context cues. The TDC and ECP datasets are recorded from a vehicle and therefore do contain VRU-related context cues, but because they are originally intended for detection the annotations have a low frame-rate, at most 5 fps. Instead, we use the dataset from our baseline DBN [4].

The proposed RNN combines the non-linear predictive abilities of RNNs with contextual features, and is trained without any additional manual annotations on the dynamic modes. Because of the relatively small size of the dataset, we do not focus on the challenge of handling missing data, but instead perform all evaluations on the smoothed data available in the dataset of [4].

The contributions of this paper are, firstly, an end-to-end approach to incorporate context cues for path prediction through an RNN, that performs competitively to the state-of-the-art even though it is trained on a small dataset. Secondly, we show that the RNN that was trained without annotated dynamic modes can learn the switch in dynamic earlier than a model with explicit annotations. Thirdly, we evaluate the advantage of additional data for each of these context cues, and show the importance of realistic training data. Our RNN is able to compute predictions for multiple objects in real time (4ms per frame when computing 50 tracks at once) thanks to the ubiquity of GPU-based neural net frameworks, in this case Pytorch [22].

III. Methodology

The goal is to predict a probability distribution over the position x at every time step t, n steps into the future. The probability distribution depends on all the previous measurements, $y_{0:t}$, and is written as $p\left(x_{t+n}|y_{0:t}\right)$.

Predicting this probability distribution is the compact description of the problem, where there are many different ways to compute the effect of the observed measurements on the predicted distribution. In this paper we model the probability distribution as a Gaussian distribution, and incorporate the measurements through an RNN.

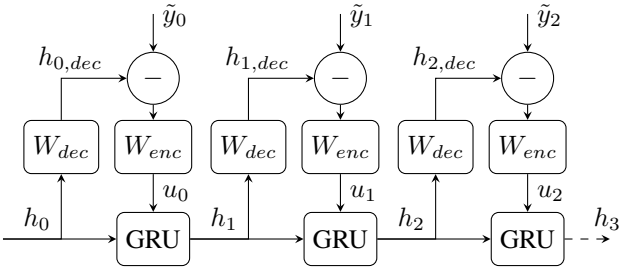


Fig. 2: An overview of how the RNN processes the measurements over time. This figure shows the incorporation of inputs over three time steps.

A. Inference

A sequence of measurements $y_{0:T}$ from time 0 to time T consists at each time step t of two-dimensional position x_t, and a vector of multiple contextual cues c_t. The data given as input to the RNN at a time step t, noted as $\tilde{y}_t$, are the difference in position between two time steps, $x_t - x_{t-1}$, and context variables: $\tilde{y}_t = [x_t - x_{t-1}, c_t]^\top$. At t_0 the difference in position is taken as zero.

The architecture of the RNN can be split up into two steps: processing the inputs over time, and the prediction. How the RNN processes inputs is shown in fig. 2. The main component is a Gated Recurrent Unit (GRU), which is used over an LSTM because of its reduced amount of parameters, something that is desirable for small datasets.

The hidden layer h_t, a vector with 32 elements, is decoded into an expected input, which is subtracted from the actual input, and the result u_t is fed into the GRU:

$$u_t = W_{enc}\left(\tilde{y}_t - W_{dec}(h_t)\right) \tag{1}$$

$$= W_{enc}\left(\begin{bmatrix} x_t - x_{t-1} \\ c_t \end{bmatrix} - \begin{bmatrix} W_{pos}(h_t) \\ W_{cues}(h_t) \end{bmatrix}\right), \tag{2}$$

where $W_{dec}(h_t) = w_{enc}h_t + b_{enc}$, a regular linear layer with w_{enc} and b_{enc} as trainable parameters. All other functions $W(\cdot)$ are linear layers as well, with their own trainable parameters.

When predicting, the signal that is fed into the GRU is computed as

$$u_t = W_{enc}\left(\mathbf{0}\right). \tag{3}$$

All future hidden states $h_{t+2}, \ldots, h_{t+n}$ are then computed as shown in fig. 3. The estimated future position $\hat{x}_{t+n}$ and covariance Σ_{t+n} is then computed as follows:

$$\hat{x}_{t+n} = x_t + \sum_{i=1}^{n} W_{pos}(h_{t+i}) \tag{4}$$

$$\begin{bmatrix} l^{[0]} & l^{[1]} & l^{[2]} \end{bmatrix}^\top = W_{cov}(h_{t+n}) \tag{5}$$

$$\sigma_1 = \exp\left(l^{[0]}\right) \tag{6}$$

$$\sigma_2 = \exp\left(l^{[1]}\right) \tag{7}$$

$$\rho = \tanh\left(l^{[2]}\right) \tag{8}$$

$$\Sigma_{t+n} = \begin{bmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{bmatrix}. \tag{9}$$

This method of computing the 2D covariance in an RNN (eqs. (6) to (9)) is taken from [17].

B. Training

The RNN is trained by minimizing the log likelihood loss of the predicted Gaussian distribution. The trained parameters are those of the GRU, the layers W_{enc}, W_{pos},W_{cues},W_{cov}, and h_0. To ensure that the output of the RNN is a consistent path, we compute the loss not just n time steps ahead, but over the entire range of 1 over n steps ahead.

Additionally, two regularization methods are used to reduce the ability of the RNN to overfit on the data. Firstly, all data is normalized. The mean and variance of $\tilde{y}_t$ in the training data are computed. The input $\tilde{y}_t$ is scaled and translated according to this mean and variance before it is fed to the RNN. The inverse of the scaling and translation is applied on the output of each prediction step in eq. (4), i.e. $W_{pos}(h_{t+i})$. The covariance matrix that is estimated by the RNN is not scaled in any way. Secondly, during training, we reset the hidden state h_t back to the initial hidden state h_0 with a probability of 5% at every time step.

IV. Dataset

The RNN described in the previous section is trained and evaluated on the tracks from the cyclist scenario in [4]. This dataset contains 51 tracks of a cyclist approaching an intersection. These are recorded with a stereo-camera setup at 16 fps from a moving vehicle that drives behind the cyclist. At the intersection, the cyclist can either continue straight or turn left. The dataset contains the longitudinal and lateral position of the cyclist in a global reference frame, as well as three additional contextual features, which are pictured in fig. 1. The first context feature is the distance of the cyclist to the intersection directly along the main axis of the road. The second feature is the time it takes for the vehicle to overtake the cyclist, if they would both keep moving with the same velocity. This feature indicates how critical the situation is for the cyclist if he wishes to turn to the left. The third feature indicates whether the arm of the cyclist is raised. This is given as a confidence in the range $[0, 1]$ as computed by a Naive Bayesian Classifier. Further details on the feature extraction can be found in [4].

Each track turning left also has one specific frame defined as the moment where the cyclists is first visibly starting to turn. The amount of frames until the turning event happens is used to temporally align all the tracks in a semantically meaningful way. This is indicated as the Time To Event (TTE). Similarly, such a point is also defined for all straight tracks. The frame TTE $= 0$ is selected as the first frame on which the cyclist is past the point on the intersection where 25% of all turning tracks have turned left.

The tracks are divided into several subscenarios, based on whether the cyclist turned left or went straight, whether the arm was raised or not, and on how critical the situation was. These subscenarios are divided into two categories: normal and anomalous behavior, based on whether it is a usual scene

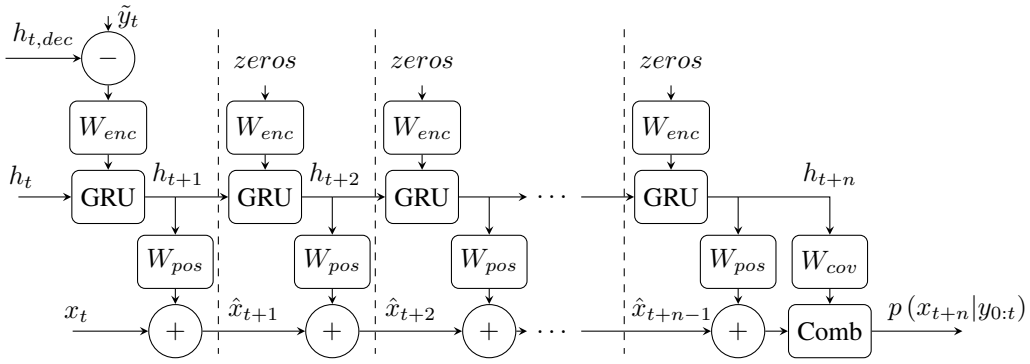


Fig. 3: An overview of how the RNN predicts $p\left(x_{t+n}|y_{0:t}\right)$ at a time step t. The layers W_{enc} and W_{pos} are shared with the temporal update process (see fig. 2). The block in the bottom right, labeled Comb, is the combination of eqs. (4) to (9).

	Subscenario		Occurrences
non-critical	arm not raised	straight/turn	6/6
non-critical	arm raised	turn	6
critical	arm not raised	straight	10
critical	arm raised	turn	7
non-critical	arm raised	straight	5
critical	arm raised	straight	4
critical	arm not raised	turn	7

TABLE I: Breakdown of the number of tracks in the cyclist dataset for the normal (above the line) and anomalous (below the line) subscenarios [4].

in real traffic. For instance, raising your arm before turning left is considered normal behaviour, but raising your arm and continuing straight is anomalous. The breakdown of the amount of tracks over each subscenario is given in table I.

Finally, some of the tracks from the dataset contain frames without position information. Because the proposed RNN has no explicit way to handle missing data, we use the smoothed tracks as described in [4] for both training and evaluation.

V. Results

We evaluate the predictive abilities of the RNN with the DBN from [4] as a baseline. We do not compare to other baselines from [4] such as the Linear Dynamical System (LDS) because those performed worse than the DBN. The RNN is, as mentioned, evaluated on the smoothed tracks, so the results from [4] have been re-evaluated on the smoothed tracks as well. As in [4], we look at the average predictive log-likelihood 16 steps (one second) into the future, around the point where the cyclist may or may not turn left: the range TTE $\in [-15, 15]$. Therefore, the loss is computed over $n = 16$ steps during training. Additionally, we compare the average Euclidean error between the predicted mean 16 steps into the future and the measurement position in the same range TTE $\in [-15, 15]$.

The models are evaluated with Leave-One-Out (LOO) cross-validation. Each RNN is trained using the Amsgrad [23] algorithm for 2000 iterations, with a learning rate of 0.0015. The RNN is implemented in Pytorch [22].

A. Ablation study

We first assess the effectiveness of the two regularization methods from section III-B, data normalization and hidden state resetting. The RNN with all three contextual features is trained and evaluated on the tracks from the normal subscenarios. With both regularization methods, the proposed RNN achieves an average prediction log-likelihood of 0.81. Without normalization, the prediction log-likelihood drops to -5.85. Without resetting the hidden layer during training, it drops to -0.61. This shows that both regularization methods help improve the accuracy of the RNN, and they are therefore used in all following experiments.

B. Baseline comparison

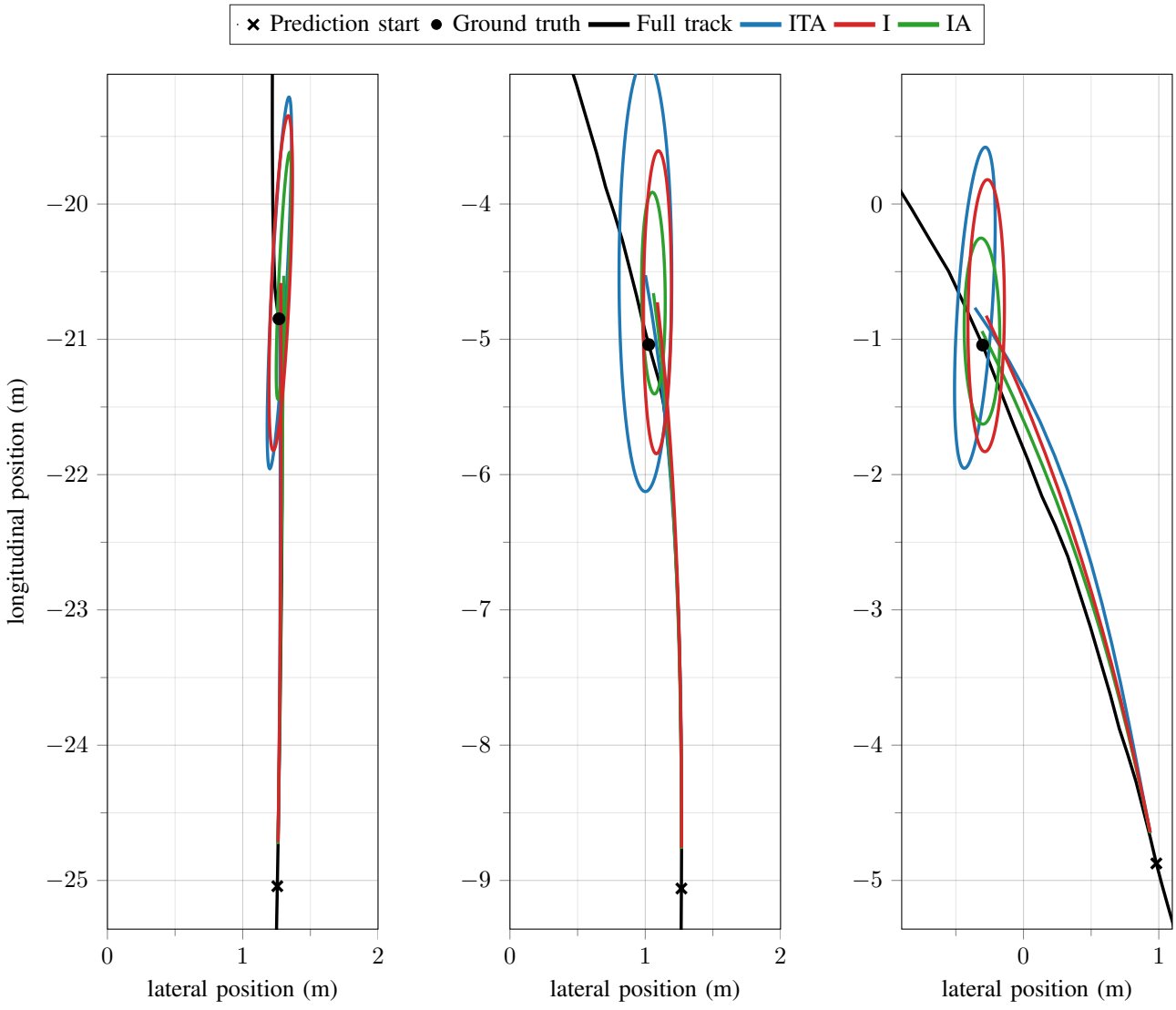
Next, we compare the prediction log-likelihoods of the RNN to that of the DBN from [4]. Here, we train and evaluate on the tracks from the normal subscenarios.

To check the effectiveness of each contextual feature, we train every combination of the three features. This is shown in table II. What contextual features are used in each RNN is indicated by one to three letters: an **I** if the RNN used the distance to **I**ntersection (static context), a **T** if the RNN used the **T**ime until the vehicle could overtake (dynamic context), and an **A** if it used the probability that the **A**rm was up (object context). The RNN that uses no additional features (i.e only the relative position between each time step) is named "No features". The RNN that uses all three features (ITA) is also referred to as the "full model" in the text.

Starting at the left of the table, it is seen that the RNN with no features already outperforms the DBN on the normal subscenarios. They also both show a large difference between the predicted log-likelihood of the straight subscenarios and that of the turning subscenarios. Furthermore, we see that the addition of one feature (the columns I, T and A) improves the likelihood even further. While combining two features does not improve the log-likelihood, the full model (ITA) does improve over the other RNNs, and attains the highest log-likelihood. The full model also attains the lowest log-likelihood on the anomalous data, which indicates that it can successfully incorporate the contextual information, as

Evaluated on	DBN	No features	I	T	A	IT	IA	TA	ITA
All normal subscenarios	−1.71	−0.42	0.68	0.40	0.51	0.47	0.57	0.36	**0.81**
Normal turning subscenarios	−2.68	−1.21	0.06	−0.58	−0.32	−0.28	−0.17	−0.70	**0.34**
Normal straight subscenarios	−0.04	0.79	1.63	1.90	1.79	1.64	1.73	**2.00**	1.55
All anomalous subscenarios	−5.94	−9.47	**0.42**	−5.54	−5.07	−1.22	−9.72	−8.76	−11.48

TABLE II: The log-likelihood of the predictions 16 steps (one second) in the future, averaged over the period TTE $\in [-15, 15]$. The models are only trained on tracks from the normal subscenarios. The best performing model is shown in **bold**. For the RNNs, the letters indicate which features were available to the RNN: **I** if the RNN used the distance to **I**ntersection, a **T** if the RNN used the **T**ime until the vehicle could overtake, and an **A** if it used the probability that the **A**rm was up.



(a) Predictions at TTE $= -71$ (b) Predictions at TTE $= -7$ (c) Predictions at TTE $= 10$

Fig. 4: Predictions over time on one specific track, for three RNNs trained on the normal subscenarios. The predictions are made from the point marked with a black x for 16 future positions. The ellipses show the 2-sigma interval of the predictions, 16 time steps ahead. The true position at the predicted time step is shown with a black dot. When the cyclist is far from the intersection, fig. 4a, the covariances are elongated and narrow. Further on, the RNNs what will happen after turning but before the turn starts (TTE $= -7$, fig. 4b). Here, the covariance becomes wider, and the predictions become slightly curved at the end. After the cyclist has turned, fig. 4c, the uncertainty is even wider.

the only difference between the normal and anomalous subscenarios is the interpretation of the contextual cues.

When comparing the average Euclidean error, the full model outperforms the other models as well, albeit only slightly: 33 cm when evaluated on the tracks from the normal subscenarios. The other RNNs with features have an error between 34 cm and 35 cm, the RNN with no features has an error of 49 cm.

A qualitative comparison of the predictions is shown in Figure 4. This figure shows the predictions of the three best-performing RNNs (ITA, I and IA) on one track at four different moments. The figure show that the predicted path of each RNN is very similar, and that all predicted paths accurately follow the true path of the cyclist.

Furthermore, the RNNs have learned to have a varying uncertainty over time. The uncertainty ellipse is very elongated early on in the track, when the cyclist is going straight (fig. 4a). As it gets closer to the point where the cyclist turns left, the uncertainty ellipse increases in width (fig. 4b). After the cyclist has turned (fig. 4c), the uncertainty in the longitudinal direction is also reduced.

C. Adding anomalous data

To analyze the impact of the dataset size, we perform the same evaluations on a larger set of tracks. For this, we train the RNNs on the tracks from both the normal and the anomalous subscenarios. This increases the available tracks during training to 50[1]. We still evaluate on the tracks of the normal subscenarios only.

The prediction log-likelihoods of this experiment are shown in table III (the top row is a repeat of the top row from table II to make the comparison easier). Many RNNs attain a higher log-likelihood on the tracks from the normal subscenarios when trained on all data than when only trained on the data from the normal subscenarios. Moreover, the best performing RNN *on the normal subscenarios* is an RNN that is trained on anomalous data as well (RNN IA). Apparently, to learn the most accurate prediction of what cyclists will do from this dataset, we must include the examples of what cyclists will not do.

We argue that this is because these models learn better motion dynamics, which are independent of the anomalous context cues. When combined with the fact that the RNN can learn highly nonlinear trajectories, this means that the anomalous tracks provide additional examples to further improve its predictions. A similar case can be made for the distance to intersection feature. The four RNNs trained on all data with the best performance on the tracks from the normal subscenario (column I, IT, IA and ITA) all have this feature in common.

The prediction log-likelihood of the full model on the normal subscenarios has decreased after training on the entire dataset: from 0.81 to 0.77. It also no longer outperforms all other RNNs. To analyze why, fig. 5 shows the predicted

[1]The total dataset is 51 tracks, but the one track used for evaluation in each LOO fold is, of course, left out for training.

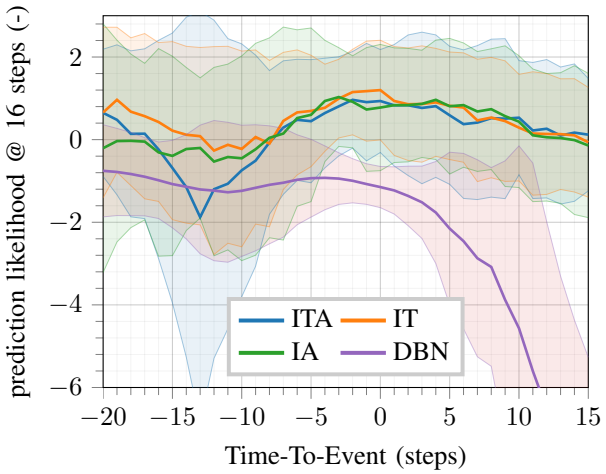


Fig. 5: The mean (lines) and one-sigma standard deviation (shaded area) of the log-likelihood of the predictions over time on all tracks in the normal turning subscenarios. The full model has a much lower prediction error than the other RNNs from TTE $= -16$ to TTE $= -8$.

likelihood over time for all tracks from the normal turning subscenarios, centered around $TTE = 0$. This graph shows the log-likelihood of a prediction made at that specific TTE, e.g. the point at TTE $= -10$ shows likelihood of the prediction for TTE $= 6$. The results are shown for the three best RNNs trained on all subscenarios, as well as the DBN trained on only the normal subscenarios (as this was the best-performing DBN). The RNNs generally outperform the DBN, especially past TTE $= 0$ where the linear model for turning of the DBN can no longer accurately predict the nonlinear turning motion of the cyclist. Furthermore, the graph shows that the full RNN has a low log-likelihood at TTE $= -13$, a moment where it predicts the location of the cyclist that has started turning left, but before having seen the cyclist actually do so. This indicates that the RNN has overfitted on the data. For completeness, the prediction log likelihood for the normal straight tracks is shown in fig. 6.

Finally, fig. 5 also shows that the RNNs increase in accuracy starting around TTE $= -10$, a moment where it makes a prediction of what happens after the turn, without having seen it. That means that the RNN is able to detect that the cyclist will turn over half a second before the annotated point of turning. We want to stress that this does not mean the RNN can see the turn over half a second before the annotator could. Instead, it shows that while the annotation of TTE $= 0$ was very consistent, it was not necessarily the moment where the dynamics change. This highlights the advantage of end-to-end learning approaches that do not depend any human-annotated information.

VI. Conclusion

In this paper, we described a Recurrent Neural Network (RNN) for predicting a Gaussian distribution over the future VRU position that incorporates various types of contextual cues and learn distinct dynamic modes. The RNN can be

Used training data	No features	I	T	A	IT	IA	TA	ITA
Normal subscenarios only	−0.42	0.68	0.40	0.51	0.47	0.57	0.36	**0.81**
Normal and anomalous subscenarios	−0.25	0.77	0.26	0.55	0.88	**0.93**	0.61	0.77

TABLE III: The effect of different training sets on the log-likelihood of the predictions *on tracks from the normal subscenarios*, 16 steps (one second) in the future, averaged over the period TTE $\in [-15, 15]$. The best performing RNN is shown in **bold**. The letters indicate which features were available to the RNN: **I** if the RNN used the distance to **I**ntersection, a **T** if the RNN used the **T**ime until the vehicle could overtake, and an **A** if it used the probability that the **A**rm was up. The best performing model *when evaluating on only the normal subscenarios*, is a model trained on anomalous data as well.

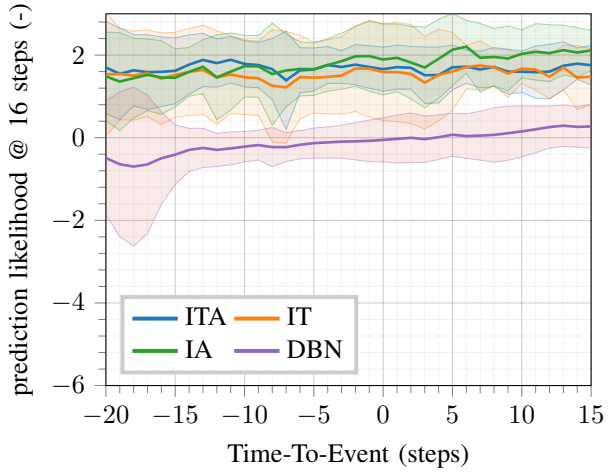


Fig. 6: The mean (lines) and one-sigma standard deviation (shaded areas) of the log-likelihood of predictions over time for all tracks in the normal straight subscenarios. All RNNs perform similar in this case, and outperfrm the DBN.

trained in an end-to-end fashion, removing the need for any ground-truth annotations. We showed that this RNN predicts the future position of the cyclist with an accuracy that is competitive with that of a method where the effects of the context cues on the dynamics modes was explicitly modeled (i.e. handcrafted). This RNN attains an average prediction error of 33 cm when predicting the future position one second into the future. Furthermore, the accuracy of the prediction scales with the amount of training data, in some cases even when the training data is not representative of the test data, and improves over training with only normative data. We consider this as a strong argument in favor of investigating methods that exploit large-scale naturalistic datasets (e.g. [21]) without the need for manual annotations of complex behaviors.

ACKNOWLEDGMENT

This work was supported by the European Community (within the PROSPECT project, grant agreement nr. 634149) and by the Dutch Science Foundation NWO-TTW (within the SafeVRU project, nr. 14667).

REFERENCES

[1] K. M. Kitani, B. D. Ziebart, J. A. Bagnell, and M. Hebert, "Activity forecasting," in *Proc. of the ECCV*, 2012, pp. 201–214.

[2] E. A. I. Pool, J. F. P. Kooij, and D. M. Gavrila, "Using road topology to improve cyclist path prediction," in *Proc. of the IEEE IV*, 2017, pp. 289–296.

[3] I. Batkovic, M. Zanon, N. Lubbe, and P. Falcone, "A computationally efficient model for pedestrian motion prediction," in *Proc. of the IEEE ECC*, 2018, pp. 374–379.

[4] J. F. P. Kooij, F. Flohr, E. A. I. Pool, and D. M. Gavrila, "Context-Based Path Prediction for Targets with Switching Dynamics," *IJCV*, vol. 127, no. 3, pp. 239–262, 2019.

[5] R. Quintero, I. Parra, D. F. Llorca, and M. Sotelo, "Pedestrian intention and pose prediction through dynamical models and behaviour classification," in *Proc of the IEEE ITSC*, 2015, pp. 83–88.

[6] V. Karasev, A. Ayvaci, B. Heisele, and S. Soatto, "Intent-aware long-term prediction of pedestrian motion," in *IEEE ICRA*, 2016, pp. 2543–2549.

[7] S. Huang, X. Li, Z. Zhang, Z. He, F. Wu, W. Liu, J. Tang, and Y. Zhuang, "Deep learning driven visual path prediction from a single image," *IEEE Trans. on Image Processing*, vol. 25, no. 12, pp. 5892–5904, 2016.

[8] E. Rehder, F. Wirth, M. Lauer, and C. Stiller, "Pedestrian prediction by planning using deep neural networks," in *IEEE ICRA*, 2018, pp. 1–5.

[9] L. Ballan, F. Castaldo, A. Alahi, F. Palmieri, and S. Savarese, "Knowledge transfer for scene-specific motion prediction," in *Proc. of the ECCV*, 2016, pp. 697–713.

[10] N. Lee, W. Choi, P. Vernaza, C. B. Choy, P. H. Torr, and M. Chandraker, "Desire: Distant future prediction in dynamic scenes with interacting agents," in *Proc. of the IEEE CVPR* , 2017, pp. 336–345.

[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. of the IEEE CVPR* , 2016, pp. 961–971.

[12] C. G. Keller and D. M. Gavrila, "Will the pedestrian cross? A study on pedestrian path prediction," *IEEE Trans. on ITS*, vol. 15, no. 2, pp. 494–506, 2014.

[13] O. Ghori, R. Mackowiak, M. Bautista, N. Beuter, L. Drumond, and F. Diego, "Learning to Forecast Pedestrian Intention from Pose Dynamics," in *Proc. of the IEEE IV*, no. Iv, 2018, pp. 1277–1284.

[14] S. Zernetsch, V. Kress, B. Sick, and K. Doll, "Early start intention detection of cyclists using motion history images and a deep residual network," *Proc. of the IEEE IV*, pp. 1–6, 2018.

[15] A. Bhattacharyya, M. Fritz, and B. Schiele, "Long-term on-board prediction of people in traffic scenes under uncertainty," in *Proc. of the IEEE CVPR* , 2018, pp. 4194–4202.

[16] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS Workshop on Deep Learning*, 2014.

[17] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013.

[18] P. Ondruska and I. Posner, "Deep tracking: Seeing beyond seeing using recurrent neural networks," in *Proc. of the AAAI*, 2016.

[19] M. Fraccaro, S. Kamronn, U. Paquet, and O. Winther, "A disentangled recognition and nonlinear dynamics model for unsupervised learning," in *NIPS*, 2017, pp. 3601–3610.

[20] X. Li, F. Flohr, Y. Yang, H. Xiong, M. Braun, S. Pan, K. Li, and D. M. Gavrila, "A new benchmark for vision-based cyclist detection," in *Proc. of the IEEE IV*, 2016, pp. 1028–1033.

[21] M. Braun, S. Krebs, F. Flohr, and D. M. Gavrila, "EuroCity Persons: A Novel Benchmark for Person Detection in Automotive Context," *IEEE Trans. on Pattern Analysis Machine Intelligence*, DOI 10.1109/TPAMI.2019.2897684, 2019.

[22] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," in *NIPS*, 2017.

[23] S. J. Reddi, S. Kale, and S. Kumar, "On the convergence of adam and beyond," in *Proc. of the ICLR*, 2018.