2018 21st International Conference on Intelligent Transportation Systems (ITSC)
Maui, Hawaii, USA, November 4-7, 2018

Accident Scenario Generation with Recurrent Neural Networks

Ian Rhys Jenkins*, Ludvig Oliver Gee*, Alessia Knauss**, Hang Yin‡, and Jan Schroeder*
*Department of Computer Science and Engineering, Chalmers | University of Gothenburg, Gothenburg, Sweden
Email: ianjenkinsresearch@gmail.com, ludvig.gee@gmail.com, jan.schroder@gu.se
**Veoneer Research, Vårgårda, Sweden, Email: alessia.knauss@veoneer.com
‡Zenuity, Gothenburg, Sweden, Email: hang.yin@zenuity.com

***Abstract*— There is a lack of approaches to derive accident scenarios automatically for the testing of Autonomous Drive systems. Current approaches that generate test scenarios do not scale due to the manual work required. Machine learning provides the possibility to automate such tasks. In this paper, an automated approach based on Recurrent Neural Networks to generate accident scenarios is presented. Based on a prototype, our approach is evaluated on temporal data from simulated in-vehicle and V2X data to automatically generate new accident scenarios. The results confirm that generated scenarios resemble the accidents that took place in an exclusive test set.**

I. Introduction

Intelligent vehicles and connected infrastructure will have an impact on consumers and wider-infrastructure alike. The efforts to increase autonomy could bring benefits like pleasurable travel, improved performance to lower emissions and new efficient business models. All of these aspects hinge on the increased *safety* that autonomy must deliver.

A critical challenge that is already prevalent with autonomous drive technology is the feasibility of testing [1]. As new features are introduced, increased testing to ensure safety is needed because these new functions replace the responsibilities once placed on the driver. So far, safety has depended on experts who must manually study accident databases to define testing scenarios. State-of-the-art approaches which are based on randomisation and model-based testing techniques are widely used but still depend on manual specification of the test scenarios [2] [3].

The automation of test specification is a necessary step to cope with the magnitude of testing needed. As such, testing and verification of safety-relevant features using computer simulations and automated scenario generation is becoming increasingly important [1]. The vehicle safety programme Euro NCAP's 2025 road map predicts that scenario-based testing, vehicle-to-vehicle (V2V), and vehicle-to-infrastructure (V2X) communication will play a major role in active safety [4].

With the availability of big data and affordable computational power, research has turned towards machine learning and deep learning where a large amount of data can be utilised to accomplish diverse tasks [5] [6] [7]. While large amounts of data from in-vehicle systems and sensors is becoming available, so far, the automotive industry has vastly focused on various tasks that involve object recognition [7]. However, data from both in-vehicle systems and also from vehicle to infrastructure should be used toward the modelling and generation of accident scenarios to deliver test data for Autonomous Drive testing. In other domains, solutions for prediction and generation have been achieved using Recurrent Neural Networks (RNNs), examples include prediction of wind speed to improve efficiency of wind turbines and the generation of handwriting text [5] [6]. RNNs are useful to model data that occurs in sequence and can be used to generate other new sequences. Furthermore, RNNs are known to outperform other models for time-series prediction, such as ARIMA and Kalman Filter in regard to accuracy [5] [8] [9]. Therefore, using RNNs seems promising for the task of accident scenario generation.

The goal of this study is to generate potential test scenarios without manual specification based on modeling accident data with RNNs, as we aim to avoid randomized alternations of existing scenarios to achieve more realistic scenario generation.

We formulate the following research question:
RQ: How can recurrent neural networks be used to generate accident scenarios? To answer this research question, we need to address two related aspects, the **representation of the input data** fed into the network and the **evaluation of the generated scenarios**. Thereby, we propose a feasible approach to be used to capture and generate test scenarios that represent new driver models as automated and V2X frameworks increase. We confirm that the scenarios generated resemble ground truth scenarios in most parameters in our evaluation of the approach based on simulator data of accident scenarios at an intersection that constitute common low velocity collisions and rarer high velocity collisions.

II. Background

Active safety systems in vehicles take pre-emptive measures in order to avoid situations where individuals inside or outside of the vehicle may be harmed. Testing these systems requires defined scenarios that have to be handled and passed by the active safety system under test.

Recent works describe how reconstructing crash and pre-crash scenarios from available accident databases can be achieved using model based techniques [10] [11], whereas state-of-art-approaches to automatically generate new scenarios are based on randomisation of parameters [3]. The first two methods require a degree of manual specification to define rules in order to specify scenarios. For the latter, while generation is automated, it still requires manual specification

978-1-7281-0323-5/18/$31.00 ©2018 IEEE

of base scenarios. Further, the actual randomised variants are not easily compared to a ground truth. Hence, it is unclear whether the randomised generations is sufficient for the testing of Autonomous Drive systems. Therefore, the automated specification of high-quality test scenarios remains the key to feasible testing.

Machine learning is an approach applied in the artificial intelligence domain to achieve a task considered too complex or too costly by manual specification or other computational means. Artificial neural networks (ANNs), refer to composite algorithms with the goal to learn a high dimensional function approximation of a task, often using linear or logistic regression. It is usually reserved when statistical analysis of a task is too arduous. Such a task could be complex object detection needed to predict a steering angle for driving using a camera feed [12].

Recent work shows how RNNs have been applied successfully for tasks such as the prediction of weather patterns and stock market prediction [5] [8]. Both studies indicate that RNNs outperform previous ARIMA models for time-series prediction. Generating sequences has also been applied by using RNNs, such as generating handwriting texts [6]. Because these methods show a viable approach to model and generate temporal data automatically while ensuring repeatability and feasibility, we apply RNNs to process available in-vehicle and V2X data to generate test scenarios.

In our prototype we use a popular architecture for an RNN computational node called Long Short Term Memory Network (LSTM) cells, described in Figure 1. Principally it contains a memory state to hold information and mechanisms for adding and discarding information during the learning process. This memory is shown as a *cell state* running through the LSTM cell and is altered through element-wise operations $\otimes$ and $\oplus$. The amount and which data to let through is selected by various gates in the cell.

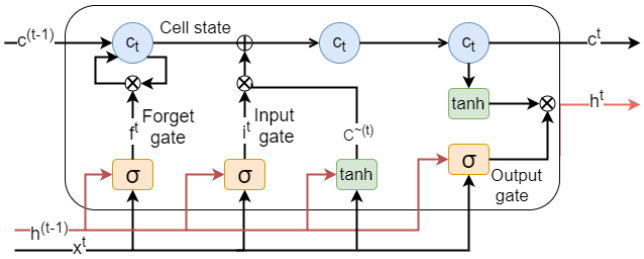


Fig. 1. Long Short Term Memory (LSTM) cell. Fundamental components of a LSTM cell are a forget gate, input gate, output gate and a cell state.

III. Methodology

As the goal of this study is to understand how to achieve automated accident scenario generation using recurrent neural networks (RNN), in this section we describe development of the prototype, data representation, and evaluation methods used that are relevant to automate the specification and generation process. We implemented our approach using the Python programming language and the Tensorflow library for machine learning. The data used in this study originates from a publicly available road traffic simulator[1]. Figure 2 shows the layout of the simulator used to collect data.

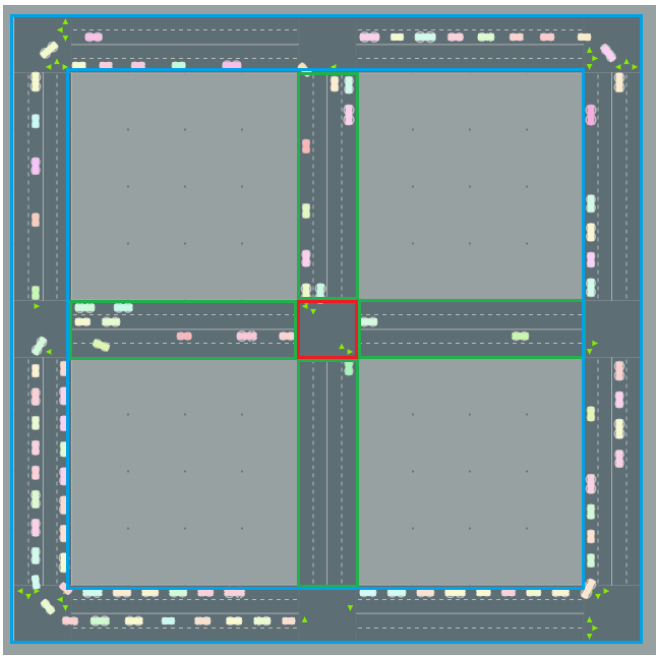


Fig. 2. Visualization of the road traffic simulator used to generate ground truth data.

The simulator integrates pre-built models such as the Intelligent Driver Model (IDM) and MOBIL that define rules and behaviour of vehicles, not built by the authors [13]. The simulator is run with 100 vehicles. We were able to record 4835 collision scenarios containing 241,750 scenes with an estimated total running-time of one week. This data is separated into training, validation, and test sets, with a respective ratio of 80/10/10 percent of the data.

A. Data Representation

From the simulator, we collect parameters that could also be collected in practice. Recent studies have shown the type of data available and what is typically used in the automotive industry. This can include GPS positioning, acceleration, speed and LIDAR point cloud data [14] [1]. We use the speed and direction of vehicles for in-vehicle data and traffic light data to represent V2X data. This data is used as input into the RNN.

B. Evaluation of the Approach

We investigate the quality of scenarios generated by comparing them with ground truth data.

There is currently a lack of evidence to support metrics to evaluate accident scenario test data. In a study predicting human trajectories, measuring average trajectories and final displacement errors is used to compare performance of different models [9]. Similarly, we are also concerned with trajectories but focus on the the speed at impact, accumulated angular change and distance travelled over an entire scenario.

[1]http://volkhin.com/RoadTrafficSimulator/

We compare how the error is distributed across a complete set of scenarios between datasets to evaluate the model on how well it learns specific qualities for accident scenarios. Given our input data and the goals we aim to achieve, we define the following metrics:

1) The distance travelled from both vehicles.
2) The total speed at collision from both vehicles.
3) The cumulative angular change from both vehicles.

IV. Automated Scenario Generation

In this section we present the scenario generation prototype in detail. A high-level design of the prototype is shown in Figure 3. Input and output data, to and from the RNNs are shown by red-dotted lines. Black lines visualize the information flow inside the RNNs. To create a scenario, the prototype has a feedback-loop where a new set of data is derived from the input, the predicted output, and an auxiliary input.

The input is called a *seed*, it contains a subset of variables $T_{vt} = \{s, d, tls\}$ where s is speed, d direction, and tls a traffic-light state. Traffic-light state refers to the traffic light data in a vehicle's heading, i.e. stop, turn right, turn left, go straight, or no data. The three variables are collected at each time step t for both of the two vehicles v. Accordingly, in total there are six variables at each time step. A seed can contain one or more consecutive subsets T_{v1},T_{v2}... T_{vn}.

Given a seed as input, the RNN produces a prediction for the vehicles at the next time-step. A prediction is a set of values $O_{t+1} = \{s, d\}$, containing values of the next speed and direction states for each vehicle. Traffic-light data for vehicles at each time-step t is fed to the network as an axillary input. Vehicles in the simulator behave accordingly to the changing states of traffic lights and given two identical seeds, modifying the traffic-light state affects the prediction from the learned model.

A. Generated Accident Scenarios

A cross-section of the types of collision scenarios captured in our test data is shown in Figure 4, represented by the speed at the impact between the vehicles.

We use the prototype to generate collision scenarios by inputting a seed and feeding auxiliary traffic-light data. To evaluate the generated scenarios in our approach, we compare them against scenarios that have the same start sequence (seed) and traffic-light data, captured in an exclusive test set (ground truth).

Figure 5 shows the x and y coordinates of three typical scenarios generated by the prototype. In the plots, generated accident scenarios are plotted in *red* and the respective ground truth scenario in blue. The left plot indicates that two vehicles were stationary in adjacent lanes at the entrance of the intersection during red traffic lights. When the lights changed to green, both vehicles accelerate and collide into each other. The middle plot depicts a scenario where one vehicle was stationary inside the intersection and the other vehicle is waiting at the entrance to the intersection due a red traffic light. Once the traffic light turned green, it accelerated

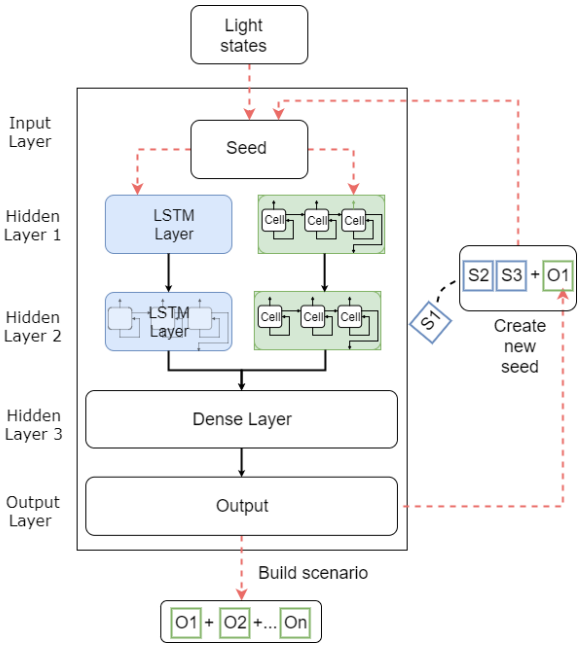


Fig. 3. Final prototype design including data flow.

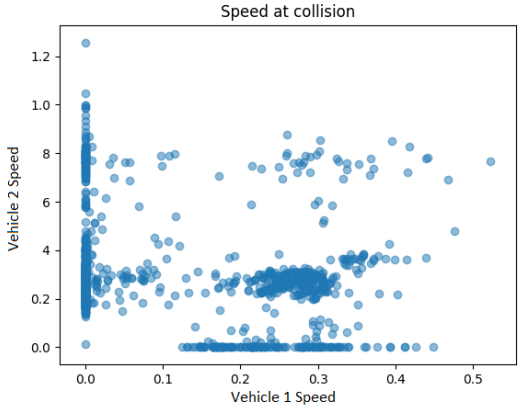


Fig. 4. The speed of the vehicles at their collisions, extracted from the ground truth data set.

and collided into the side of the stationary vehicle. The right plot indicates a high speed collision. One vehicle was stationary at a red traffic light and then accelerated across the intersection when the traffic light turned green. A second vehicle, travelling at a higher speed enters the intersection, colliding with the rear of the other vehicle.

The predicted scenarios shows how different types of accident scenarios are generated. This includes varying speeds, different traffic-light situations and lanes at the intersection. Using a seed of three sequential time-steps, the prototype is also able to generate long sequences of the pre-crash scenario by using its predicted values and an auxiliary input via the feed-back loop.

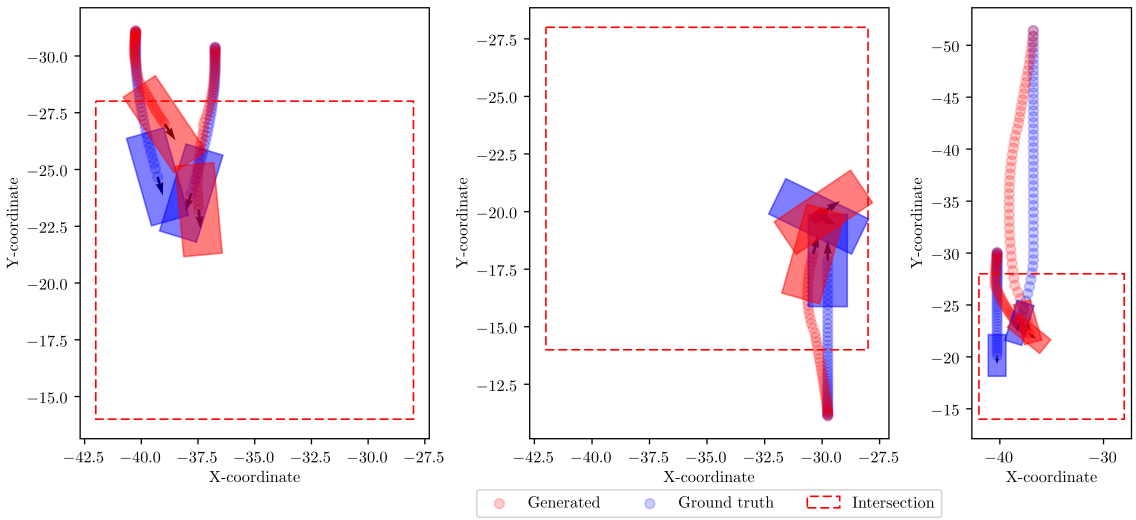


Fig. 5. Generated versus ground truth scenarios of two slow moving vehicles (left), one slow moving and one stationary vehicle (middle), and one fast moving and slow moving vehicle (right). The distance between two points indicates the distance travelled per frame; the closer the points, the lower velocity.

B. Input Data and Training

The representation and quality of input data is the most important aspect of successful machine learning [15]. To attain the best possible scenario generations we had to decide on input and training parameters. These parameters were the length of an input sequence, whether traffic light data was visible to vehicles whilst travelling inside an intersection and the configuration of the neurons within the network. They are collected in Table I, labelled as "Time-steps", "TLS" (Traffic-Light State) and "Network", respectively. Modifying these parameters affected the error between predicted values and ground truth, measured as mean squared error (MSE).

TABLE I
MEAN SQUARED ERROR VALUES PRODUCED BY MODIFYING INPUT PARAMETERS AND THE NETWORK CONFIGURATION.

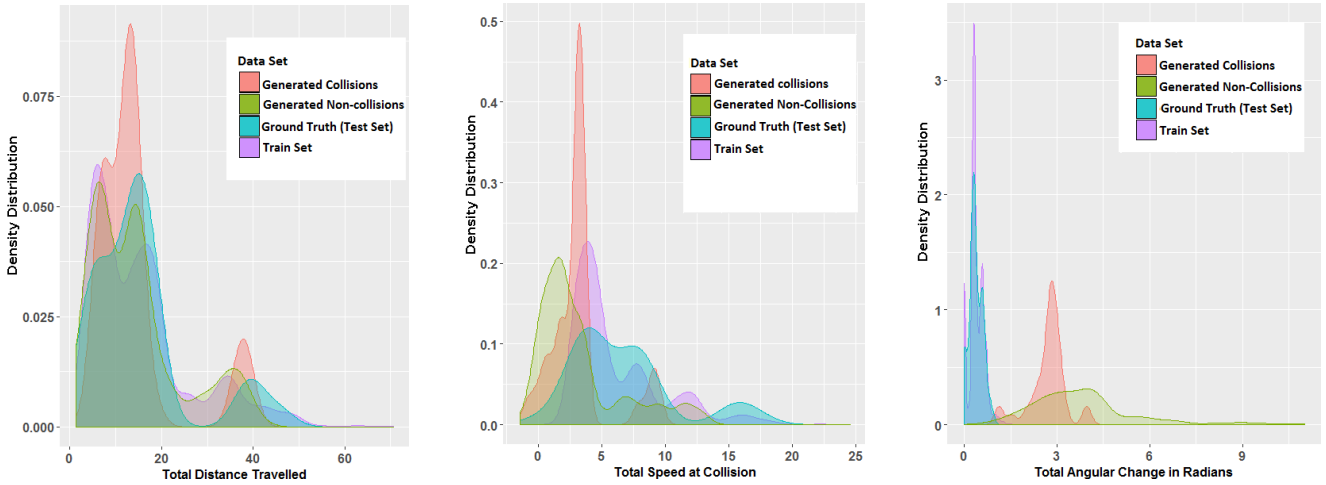
Time-steps	1	2	3	4
MSE	3.7181	2.0425	**1.6643**	1.9249
TLS	In Intersection	Not in Intersection		
MSE	5.50564E-5	**4.356E-5**		
Network	Single	Parallel		
MSE	4.41E-6	**4.0E-6**		

For the length of the input sequence, the lowest error recorded was at 3 time-steps and using more than 4 time-steps gave an overall rise in error. Traffic light states affected vehicles behaviour during the lead up to an intersection but removing TLS of vehicles in transit inside the intersection boundary, shown as "Not in Intersection" in Table I, reduced the MSE from 5.50564E-5 to 4.356E-5. The configuration of neurons used in the prototype was derived through iteration during training to find the smallest amount of neurons per layer with the largest depth where the model could converge to the lowest MSE. This is because in general more neurons per layer lead to over-fitting and more layers offer better feature extraction but are more difficult to train. The lowest error was achieved with a structure containing 32 neurons in the first layer and 64 in the second layer with the total number of $N = 96$ neurons. To reduce over-fitting, the neurons were divided into two networks (parallel network). This network consists of two RNNs connected in parallel with 16 neurons in the first layer and 32 neurons in the second. The parallel network gave a lower MSE value and converged equally well despite having less neurons per layer for each sub-network and therefore was used to constrain the model to improve generalisation.

V. Evaluation

During training we measure the MSE of a predicted output O_{t+1} and the ground truth. While this works as a training tool, it does not allow for testing the quality of the complete generated scenarios according to the evaluation criteria **1**, **2**, and **3** described in Section III. To evaluate these criteria, we use kernel density estimation (KDE) highlighting how generated scenarios from the test set are distributed in comparison to the ground truth against each criteria. The plots in Figure 6 contain four distributions: Generated scenarios with collisions in *red*, generated scenarios without collisions in *green*, the ground truth distribution of both successful and non successful collision in *blue*, and a sample distribution of scenarios from the training set in *purple*. The peaks indicate a higher distribution of scenarios at the value for the examined variable. Accordingly, the distribution curves can be compared by how well they overlap.

Figure 6a shows the distribution of the total distance travelled by the vehicles in a scenario (evaluation criteria **1**); Figure 6b, the combined speed of the vehicles at the point of impact in a scenario or at the end of a scenario if no collision took place (evaluation criteria **2**); and Figure 6c, the total accumulated angular change by vehicles in a scenario (evaluation criteria **3**).



(a) Distribution of the total distance travelled by vehicles in a scenario

(b) Distribution of speed at point of impact of vehicles in a scenario

(c) Distribution of accumulative angular change of vehicles in a scenario

Fig. 6. Kernel density plots (KDE) for the three evaluation criteria distance, speed, and angular change.

We found that the distributions fall in the ranges of the ground truth for the distance travelled and the speed at collision criteria (**1** and **2**). We can see that several peaks and troughs, which may relate to nuances that different vehicles travel in the scenarios are also captured by our model. In Figure 6a, higher distributions between the 10 to 15 distance units for successful scenarios indicate over-fitting. These kinds of observations are difficult to see using an average displacement metric because it is not made explicit. Plotting kernel densities during learning allowed for an understanding of which qualities are being learned and was useful to fine tune learning parameters.

The KDE plot of accumulative angular change shows a distribution skewed to the right, resulting in a higher angular change than desired. This is mainly due to the prediction of continuous values where an error propagates in each prediction such that predicted stationary vehicles tend to spin in one direction. To mitigate this error, the output prediction could be quantised. Therefore by taking this into account, for the generated collision plot the peaks and troughs are consistent with the ground truth.

Addressing the plots together, we can summarise that for collision plots, the distance and speed of impact are close to the ground truth. Even with the accumulated angular change (spin) error that occurs, we can understand that collisions can still occur because overall they stay in close proximity to the ground truth vehicle. For non-collisions we can understand that these generated scenarios are not always complete misses and therefore we should not discount all non-collisions. Non-collision scenarios include vehicles that have covered appropriate distances, speed at impact but minor deviation in angle resulting in no contact between vehicles at the end of the scenario. Ultimately, the alignment of the generated plots and ground truth plots delivers an increased generalised model, one that increases the accuracy of generated scenarios over different environments.

VI. Discussion

The results show that our approach using RNNs is capable of generating new accident scenarios. These new scenarios resemble their respective ground truth scenario. Hence, we know that these generations have not been randomly generated.

Our results show that the variables speed, direction, and the state of traffic lights were enough to produce valid results. The use of x and y coordinates was not needed as it yielded no improved results. Only a small number of frames of data were needed to make accurate predictions; beyond three frames, the accuracy decreased. This is likely because other variables changed state and influenced the outcome which is not captured by the input data. The effect of removing the traffic light data while a vehicle is inside an intersection suggests that some variables become an influencing factor under certain conditions and noise in others.

Because in general having more neurons per layer can lead to over-fitting, it is important to constrain data to find a robust solution. We used a parallel network and found it to be more effective than a single network.

The evaluation metrics plotted using kernel density estimation helped to describe the quality of the generated scenarios in a quantifiable manner. It also became apparent that they could be used to understand and visualise the learning process and thereby help to reduce problems such as over-fitting which should addressed in future work

A. Alternative Approaches

While there are numerous approaches to model time-series data, RNNs have been shown to outperform these other methods. One approach using RNNs to model time-series data of pedestrian movement proposes to use one RNN network per trajectory. Our approach also used more than one RNN network which we found beneficial, however our configuration is different. In general, adversarial networks where two networks compete have been shown to improve

accuracy of a desired model but was not in scope of this paper (cf. [16]). In all cases the input data and how it is represented is the key component to any method and network structure.

B. Threats to Validity

Actual crash data was not available to the researchers. To compensate, we used a sourced simulation with relevant driver models to collect appropriate accident data. While the error of the angular distribution is skewed, this could be mitigated by quantising outputs before feeding them back into the network.

VII. Conclusion

To address current feasibility challenges of testing autonomous vehicles, automated specification of test scenarios is a necessary task due to an increased level of testing needed to prove that cognitive systems of autonomous vehicles can act accordingly. To reach this goal, we have investigated how recurrent neural networks can be applied to generate accident scenarios from available simulated in-vehicle and vehicle to infrastructure data.

Our approach demonstrates successful generation of accident scenarios using recurrent neural networks. By doing so, it replaces the manual specification of defining scenarios by automatically learning accident data. The results show, the prototype captured the characteristics of the accident data set and generated low and high velocity collisions with an good level of accuracy.

Generating accident scenarios requires accurate prediction of multiple actors along a time line which increases complexity. Our approach demonstrated two trajectories simultaneously generated according to an external environment. The level of accuracy produced by the prototype; evaluated by the distance travelled, the speed at impact, and the accumulated angular change set a promising outlook for future work.

Acknowledgement

The authors would like to thank the Chalmers University of Technology and the University of Gothenburg for the Lars Pareto grant. The authors would also like to thank Certaincy AB, Sweden, as co-sponsors to this study. A part of this study has been conducted while Alessia Knauss and Hang Yin were affiliated with Chalmers University of Technology, Gothenburg.

References

[1] A. Knauss, J. Schroeder, C. Berger, and H. Eriksson, "Paving the roadway for safety of automated vehicles: An empirical study on testing challenges," in *2017 IEEE Intelligent Vehicles Symposium*, June 2017, pp. 1873–1880.

[2] H. Altinger, F. Wotawa, and M. Schurius, "Testing methods used in the automotive industry: Results from a survey," in *Proceedings of the 2014 Workshop on Joining AcadeMiA and Industry Contributions to Test Automation and Model-Based Testing*. ACM, 2014, pp. 1–6.

[3] S. Khastgir, G. Dhadyalla, S. Birrell, S. Redmond, R. Addinall, and P. Jennings, "Test scenario generation for driving simulators using constrained randomization technique," SAE, Tech. Rep., 2017.

[4] "Euro NCAP 2025 Roadmap- Vision to Zero ," https://www.euroncap.com/en/for-engineers/technical-papers/, accessed: 2017-10-19.

[5] Q. Cao, B. T. Ewing, and M. A. Thompson, "Forecasting wind speed with recurrent neural networks," *European Journal of Operational Research*, vol. 221, no. 1, pp. 148 – 154, 2012. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221712001920

[6] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation," *arXiv preprint arXiv:1502.04623*, 2015.

[7] D. Tomè, F. Monti, L. Baroffio, L. Bondi, M. Tagliasacchi, and S. Tubaro, "Deep convolutional neural networks for pedestrian detection," *Signal Processing: Image Communication*, vol. 47, pp. 482–489, 2016.

[8] A. A. Adebiyi, A. O. Adewumi, and C. K. Ayo, "Comparison of arima and artificial neural networks models for stock price prediction," *Journal of Applied Mathematics*, vol. 2014, p. 7, 2014.

[9] A. Alahi, K. Goel, V. Ramanathan, A. Robicquet, L. Fei-Fei, and S. Savarese, "Social lstm: Human trajectory prediction in crowded spaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 961–971.

[10] J. J. Ference, S. Szabo, and W. G. Najm, "Objective test scenarios for integrated vehicle-based safety systems," in *20th International Technical Conference on Enhanced Safety of Vehicles (ESV), Lyon, France*, 2007.

[11] F. Spitzhüttl, H. Liers, and M. Petzold, "Creation of pre-crash simulations in global traffic accident scenarios based on the iglad database," in *FAST-zero'15: 3rd International Symposium on Future Active Safety Technology Toward zero traffic accidents, 2015*, 2015.

[12] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, "End to end learning for self-driving cars," *CoRR*, vol. abs/1604.07316, 2016.

[13] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Journal of the Transportation Research Board*, no. 1999, pp. 86–94, 2007.

[14] X. Wang, D. Zhao, H. Peng, and D. J. LeBlanc, "Analysis of unprotected intersection left-turn conflicts based on naturalistic driving data," *arXiv preprint arXiv:1702.00135*, 2017.

[15] S. Kotsiantis, D. Kanellopoulos, and P. Pintelas, "Data preprocessing for supervised leaning," *International Journal of Computer Science*, vol. 1, no. 2, pp. 111–117, 2006.

[16] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.