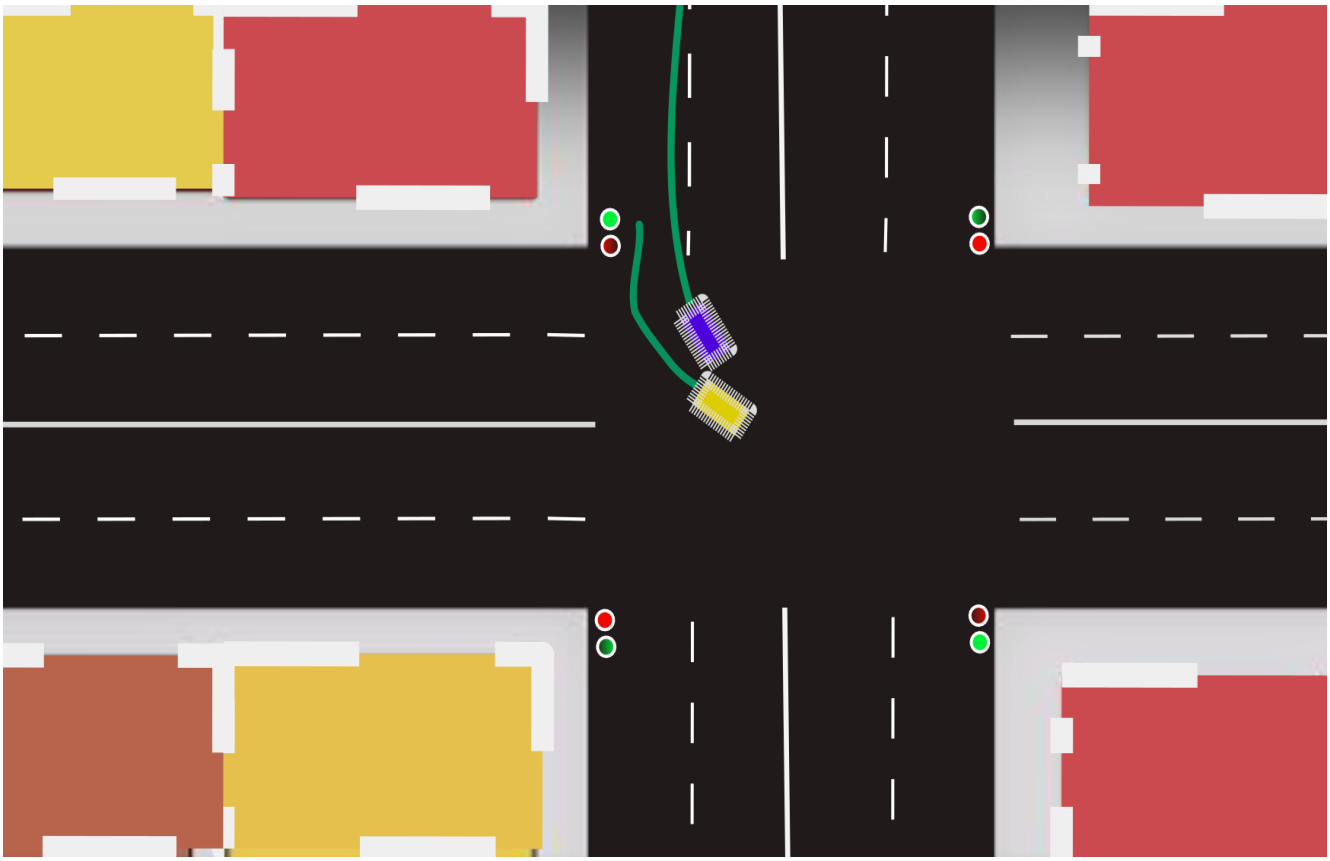

An approach to automate accident scenario generation using recurrent neural networks

Bachelor of Science Thesis in Software Engineering and Management

LUDVIG OLIVER GEE
IAN RHYS JENKINS

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

{An approach to automate accident scenario generation using recurrent neural networks}

{LUDVIG OLIVER. GEE , }
[IAN RHYS. JENKINS,]

Examiner: {MICHEL R.V. CHAUDRON}

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

[Cover:
An accident scenario generated by the Generation Model in Fig. 10 on page 7. It shows how an accident occurred between a low speed and high speed vehicle at an intersection, from given starting parameters. An animation can be viewed from: https://sites.google.com/view/activesafetyrnn/home]

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Department of Computer Science and Engineering
UNIVERSITY OF GOTHENBURG
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2017

An approach to automate accident scenario generation using recurrent neural networks

Ludvig Oliver Gee Department of Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden
ludvig.gee@gmail.com
Ian Rhys Jenkins Department of Computer Science and Engineering
University of Gothenburg
Gothenburg, Sweden
ianjenkinssem@gmail.com

***Abstract*—There is a need to improve the test procedure of Active Safety Systems through the automation of scenario generation, especially accident scenarios that are critical for testing. The purpose of this thesis is to provide an approach to automate the test generation process using machine learning. We use a recurrent neural network, applied in other domains to related problems where temporal data needs to be modelled for the generation of accident scenarios. We build a dataset of accident scenarios that occur at an intersection in a road traffic simulator and use it to train our model. We deliver an approach by testing different model parameters and input features and show generated accident scenarios in comparison to ground truth scenarios. We evaluate the quality of our generated accident scenarios through a set of metrics which we introduce in the paper.**

***Keywords*-Machine Learning, Recurrent Neural Networks, Active Safety Systems, Scenario Generation, Testing, Time Series Prediction.**

I. Introduction

The automotive industry and consumers are preparing for an autonomous world where vehicles will navigate seamlessly through a concoction of unfolding events. Ultimately this leads to a resolution of trust between humans and the systems designed to keep them safe. Today's vehicles have powerful on-board computer systems and amongst them are Active Safety systems which are responsible for decisively mitigating accidents and minimising damage [1]. State of the art systems in vehicles include safety critical Advanced Driver Assistance Systems(ADAS) and Autonomous Drive(AD) which have to meet stringent standards. The current verification and validation process involved in meeting these standards must be run at different levels of development and crucial among these tests is how integrated vehicle systems respond to reconstructed events. The binding of vehicle autonomy, sophisticated on-board devices and increased integration of a vehicle's systems requires more complex reconstructed scenarios for testing to safely deploy autonomous vehicles that are still feasible.

For the Active Safety domain, the need to attain realistic test scenarios within a reasonable effort persists. Much is done through manual analysis taken from accident databases [2] and even though model based testing techniques are widely used, manual specification of test scenarios is still the norm [3]. State-of-the-art vehicles use a wide range of sensors to capture data with, utilising this data to model and derive tests would be beneficial however the state space is so large that this makes it impossible to manually analyse. Real life scenarios involve external actors such as pedestrians, traffic signals, obstructions to lines of sight and behavioural patterns of other drivers. Statistical models are useful for modelling data but capturing models that are non-linear models or without prior specification is better achieved by applying Artificial Intelligence (AI) methods [4] [5]. Euro NCAP 2025 roadmap - Vision to Zero suggests the domain will become centred towards scenario based testing and that *vehicle to vehicle* and *vehicle to infrastructure* (V2x) communication will play a major role in active safety [6] . To recreate complex scenarios which becomes necessary to test an autonomous infrastructure of vehicles, automated and other capable methods are needed. It is important to understand that there are many tools for automated test *execution* and Continuous Integration but not for automatic test *generation*. Our paper provides an approach which includes a prototype that uses artificial intelligence to generate accident scenarios by modelling data taken from a simulator. We also provide a set of metrics to evaluate generated scenarios. Research by A. Knauss et al. [7], on testing challenges for safety of automated vehicles, mentions that a promising strategy for extracting test cases from real-world events is possible with the application of Artificial Intelligence(AI) on big data.

Machine Learning is an approach within the Artificial Intelligence domain. "A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E." -Tom Mitchell (1997) [8]. The premise is that data can be fed into a program in such a way that a program's performance of a task is improved through the experience of information. More data improves the scope for understanding and revision of that information increases knowledge.

Artificial Neural Networks (ANN) are part of a family of machine learning algorithms such as Support Vector Machines and Naive Bayes. A distinction between a traditional program and a program that is an ANN is that a task is not executed

by an explicit set of instructions. Instead an ANN is an environment of neurons with constraints (connections between neurons) where a task may be learnt.

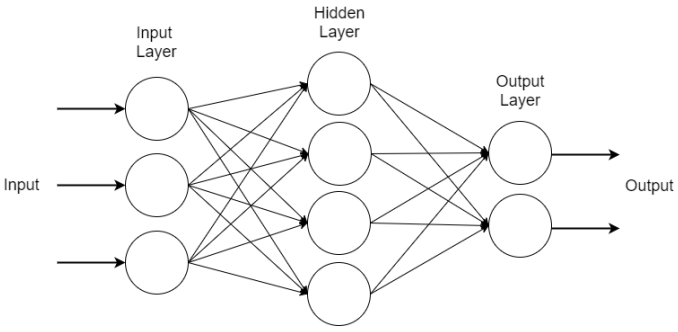

Fig. 1. Example of Forward-Feed Network

So called Deep Learning has revitalised the neural network domain with a combination of layered networks, big data, and computational power previously unavailable. Two important ANN structures exist: forward-feed networks and recurrent neural networks (RNN). For forward-feed networks data moves one way between neurons (shown in Fig. 1), whilst in recurrent networks (shown in Fig. 2) data can flow in cycles, not only restricted to forward connections.

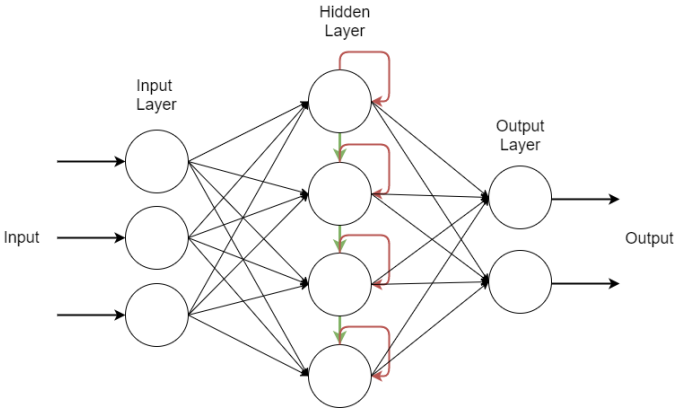

Fig. 2. Example of Recurrent Network

Machine learning is being applied to range of applications from image classification in applications like Snapchat [9] to regression for prediction of weather patterns [5]. Forward-feed networks such as convolution neural networks are being researched for uses in image recognition in the automotive industry [10] and increasingly RNNs are explored in domains where temporal information is important [11] [12].

Vehicle manufacturers together with suppliers and a wider infrastructure have set a goal for zero traffic fatalities. Being able to accurately construct detailed accident scenarios will be crucial because as vehicle systems become more complex with rich features and because these new features transfer control from humans to automated systems, a complete certainty of these new features under test must be guaranteed. Once deployed, these new systems will continue to reduce the number of incidents whilst a potential for new types of risks will develop. Resources for collecting and methods for deriving test scenarios requires re-thinking and in this paper we explore the practical application of using an recurrent neural network, a machine learning approach to generate test scenarios which leads to our research question:

RQ: How can Recurrent Neural Networks be applied to achieve automated accident scenario generation for Active Safety system testing?

We verify our approach through testing different model parameters and features of our dataset. We evaluate how accurately our model can generate collision scenarios at an intersection by measuring the quality of scenarios by a set of metrics.

Developing a model that could automate the generation of scenarios could reduce the errors caused and the effort needed to model each specific actor involved through manual specification. Using such a model would help automate test scenario generation in a similar approach to S. Khastgir et al. [13] without the need of actively defining constraints and dependencies between actors in a given environment. Such generated scenarios could be rich in detail and utilise simulation or augmented reality to deploy tests, reducing costs towards accomplishing feasibility.

As the scope of possible scenarios is large we narrowed our focus on modelling accident scenarios at an intersection. Our goal is to provide an approach that could lead to an end-to-end solution which can take real value data, such as data from on-board systems in vehicles, to automatically generate feasible test scenarios.

II. Background

A. Active safety systems and testing

Active safety systems in vehicles are those systems that take preemptive measures in order to avoid situations where individuals inside or outside of the vehicle may be injured or in the worst case killed. An example of this is a forward collision warning system where the vehicle detects a slow or stationary object ahead of the vehicle that might cause a collision. Depending on how the system is built it can either alert the driver of the upcoming hazard or trigger another system that takes control of the vehicle to avoid an imminent collision.

To test one of these systems there needs to be a defined scenario that the system has to handle and pass. For this paper we will use the following definition of a scene and scenario:

> "A scene describes a snapshot of the environment including the scenery and dynamic elements, as well as all actors' and observers' self-representations, and the relationships among those entities. Only a scene representation in a simulated world can be all-encompassing (objective scene, ground truth). In the real world it is incomplete, incorrect, uncertain, and from one or several observers' points of view (subjective scene)." [14]
>
> "A scenario describes the temporal development between several scenes in a sequence of scenes. Every scenario starts with an initial scene. Actions and events as well as goals and values may be

specified to characterise this temporal development in a scenario. Other than a scene, a scenario spans a certain amount of time." [14]

Other terms used in this paper are :

- "Scene" and "frame" where a scene is the metadata within a frame.
- The words "timestep" and "frame" are interchangeable.
- "Accident scenario" and "collision scenario" are used interchangeably as well.
- The term "seed" is used throughout the Results and Discussion sections refers to input values that the model requires to produce an output.

.

B. Machine Learning with ANN

With the notion of learning through experience of information, knowledge is stored through iterative change of an abstract memory state: the learning parameters in artificial neurons. Neurons can have different and complex cell structures and are based on the neuron shown in Fig. 3.

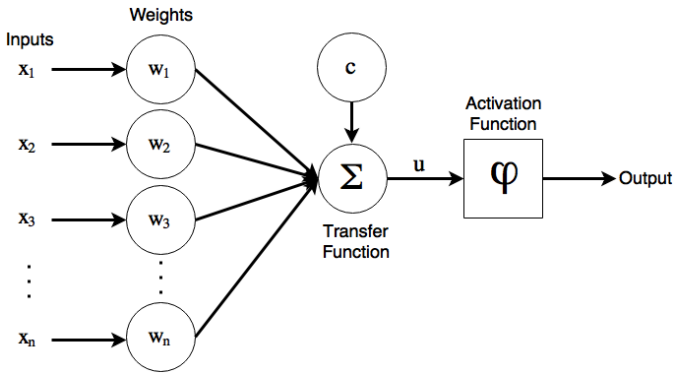

Fig. 3. Simple artificial neuron. C is a bias function

Simply put, a neuron takes an input value for some task and computes an output according to a computation with an initial randomised learning parameter value. We expect some error since this learning parameter is random and so the parameter value is altered to some degree to potentially reduce the error. A task is tried again and again until some level of satisfaction or exhaustion is reached. Intuitively it is not dissimilar to the way in which humans learn. As a child we learn to use utensils to eat, awfully at first with lots of errors but with practice we learn how to manage the complex problem! Likewise, artificial neurons can model complex tasks by intricate connections between neurons.

A neuron shown in Fig. 3, also referred to as a cell, contains learnable parameters w_i and where a nonlinear activation function is applied to a transfer function, the following two equations describe the process. An activation function:

$$h = g(u) \tag{1}$$

where g is commonly a logistic sigmoid or hyperbolic tan function and h by convention is used to denote a hidden output, shown as *output* in Fig. 3.

And a transfer function :

$$u = \sum_{i=1}^{n} W_i x_i + c \tag{2}$$

where x_i are inputs, w_i weights and c a bias function.

The activation function in equation 1 outputs a squashed value which becomes the input into another neuron. Connections between neurons and layers of neurons complete a network through which a learning task takes place.

A goal of learning is to generalise. Data that a model has never previously seen during learning should output a value accurate to some degree. Often, causes for inaccuracy come from underfitting or overfitting. Underfitting occurs when a model has not been able to learn data well enough, overfitting occurs when a model has learnt the pattern of a training set so precisely that predicting any new unique data is now inaccurate to some valid degree. We say that neurons learn weights W by reducing a margin of error through updating its learnable parameters.

Models can be taught to generalise data by either supervised, unsupervised or semi-supervised learning. In supervised learning a model is provided with a target value to measure its own predictions against in order to adjust its learning parameters to achieve a better prediction. Unsupervised learning implies that no target value is provided and a typical application for this is clustering where the target value is not known. Semi-supervised learning may include target values on some data but not on others. There is no formal definition but in our research we use supervised learning and regression to predict vehicle states.

To put into context neurons and learning Fig. 4 illustrates an entire flow, forwards (black) to prediction and backwards (blue) updating learning parameters. The network is constructed as a directed graph.

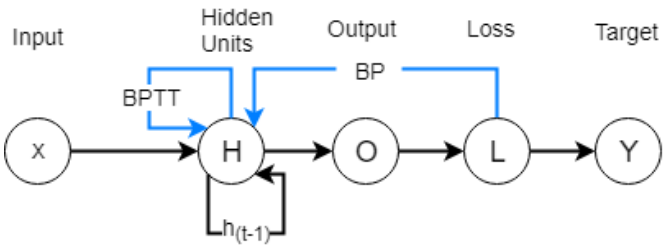

Fig. 4. Forward pass (black) and backward pass (blue) for a recurrent neural network

During a forward pass through a network, input values X are passed through some function in hidden units H where the output values are measured against target values T to compute an error. This is refered to as a loss function such as the Mean Squared Error (3):

$$MSE = \frac{1}{n} \sum_{t=1}^{n} e_t^2 \tag{3}$$

Remembering that a network learns through updating its learning parameters W seen in Fig. 3 and that a network is a

set of directed nodes show in 4, then adjusting learning parameters values can be done through moving through the nodes in a graph. A backpropagation algorithm can be used for this task and in practice is calculated using a computation graph. Within each neuron there is a set of nodes which represent a function $f(x, \theta)$ and edges that are the output values. Backpropagation works by computing the partial derivative of the output from the loss and recursively flowing backwards. The local gradient from each node function in a computational graph is computed and the partial derivative of composite functions can be derived using the chain rule, that if $f = qz$ and $q = x + y$ then the partial derivative of x can be found by:

$$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial q}\frac{\partial q}{\partial x} \tag{4}$$

This final step to learning is an optimisation problem, how to adjust the learning parameters so that the error produced by the next forward pass is minimised. To solve this a Gradient Descent algorithm (5) is applied to minimise the cost function of some high dimensional space:

$$\Delta W^n = -\alpha \frac{\partial L}{\partial w^n} \tag{5}$$

where α is a learning rate between 0 and 1.

Using the derivative of the error function, the gradient of the steepest slope is found from a randomised starting point in the high dimensional space. The amount in which we move down along the gradient towards a local minima is set by a hyper parameter, a learning rate shown by the paths in in Fig. 5[1]. At this new point the learning parameters can be updated for the next forward pass. This set of subtle processes is a generalisation of how learning in supervised deep networks is accomplished in practice and many optimisation techniques are available for computing nodes and moving in a high dimensional space.

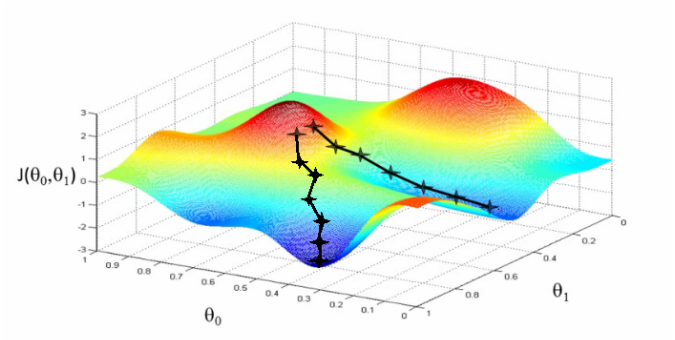

Fig. 5. Gradient descent from two randomised points

C. Recurrent Neural Networks

Recurrent Neural Networks(RNN) are a category of ANNs which can take a sequence of values as input to output a hidden state. A hidden output of recurrent cell is described by the equation:

$$h_t = f(h_{(t-1)}, x_t; \theta) \tag{6}$$

where $t \in T$ and T is a sequence of timesteps, h_t is a hidden state, x_t an external input with θ as parameters. Fig. 6 illustrates an RNN cell with hidden outputs as a cyclic graph(left) and depicted as an unfolded computation graph(right). For RNNs the special case of backpropagation of cyclic graphs can be solved by unrolling a cyclic graph into finite steps shown in Fig. 6. This is equivalent to a forward-feed network where backpropagation can be applied.

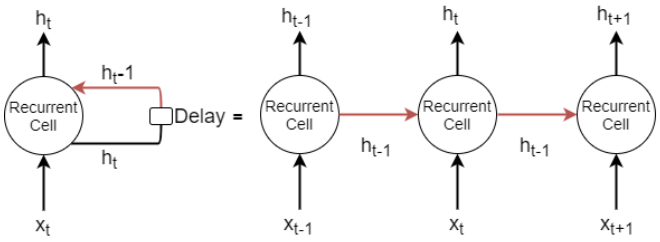

Fig. 6. RNN cell with hidden outputs(left). Unrolled computation graph(right)

Forward-feed only networks output values from one neuron layer into another in one direction. Each input to output through the network is independent of each other. For RNNs, the sequence of inputs are not independent. The number of cycles in the graph corresponds to the number of sequences which the network is fed which we refer to as a timestep. A timestep could be seconds, hours, days or an abstract value.

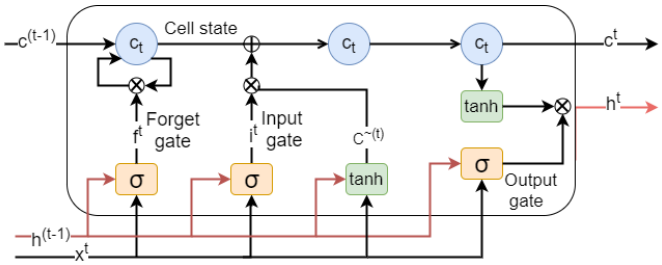

Fig. 7. Long Short Term Memory(LSTM) cell

A popular cell architecture for Recurrent Neural Networks is Long Short Term Memory Networks (LSTM) that have a more detailed cell structure than a regular RNN described above. Alex Graves (2012) [9] gives a detailed description of LSTMs, here we describe useful concepts for understanding our methodology. An LSTM cell has a memory state that can add and discard information during learning. In a regular RNN new information is learned but past information is eventually lost as there is no mechanism for storing, adding, and discarding information. LSTM networks introduced by Sepp Hochreiter and Jürgen Schmidhuber help towards rectifying the problem of losing information [15].

The fundamental components of an LSTM cell are: a forget gate, input gate, output gate and a cell state that form four layers illustrated in Fig. 7. A *forget gate* selects which data

[1]https://www.analyticsvidhya.com/blog/2017/03/introduction-to-gradient-descent-algorithm-along-its-variants/

to discard, an *input gate* selects which data will be updated and what new cell state values are to be added. The *cell state* is altered through element-wise operations $\otimes$ and $\oplus$ and the *output gate* decides what to output in relation to the input which will be combined with it's cell state to output a new hidden state h_t. Equations (4-8) describe the hidden outputs H of an LSTM cell as a composite function [16]:

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf} + b_f \quad (7)$$

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{c_i}c_{t-1} + b_i) \quad (8)$$

$$c_t = f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (9)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \quad (10)$$

$$h_t = o_t \tanh(c_t) \quad (11)$$

where f, i, o and c are the activation outputs of forget, input gates, output gates and cell state and h is the hidden output from a cell. W_xf, W_xi, W_xc, W_xo are the input weight matrices for each gate, h_f, h_i, h_c, h_o are hidden input matrices, and t is the timestep. b are bias terms not included for clarity.

III. Related Work

Within the Active Safety domain problems exists to attain realistic test scenarios within reasonable effort. The following describes the state-of-the-art rationale and techniques of data collection, analysis and modelling for testing within the domain.

In-vehicle systems are used to collect data about behaviour in the field [17]. The data collected can include GPS positioning, acceleration, speed, and LIDAR point cloud data [18] [19]. Data is also collected from driving simulators that use human participants set in a virtual environment where they encounter different traffic situations. Their subsequent actions, steer angle, brake pedal depression and gaze direction can then be recorded [20]. To determine when and what data to analyse there are metrics such as surrogate safety measures e.g time to collision between two vehicles, that can be used as indicators of interesting events to analyse [21]. The data collected by these systems is then in turn used to model actors in simulators or create test scenarios. Scenario generation for Active Safety Testing is widely derived through manual specification. This has been identified as a major challenge amongst industry practitioners as the complexity of scenarios increases [19]. This is especially true as systems move towards autonomous drive. JJ Ference at al. [22] present a method for creating objective test scenarios based on a crash database. They analyse the most common scenarios and provide metrics that can be used to reconstruct those scenarios. F Spitzhüttl at al. [23] have developed a methodology of creating scenarios (pre-crash simulations) based on the iGLAD accident database[2]. As the iGLAD dataset was at times incomplete they had to compensate and introduce assumptions in order to provide a greater yield of scenarios however reducing accuracy and reliability. A quality criterion was introduced as well in order to try to address inaccuracy and reliability of the created scenarios.

The use of a constrained randomisation technique, to generate scenarios for a driving simulator, is presented by S. Khastgir et al. [13] in order to derive possible scenarios and test cases based on a given use-case.

Specific topics are studied independently such as bicycle to car collisions [24], light to heavy vehicles [18] and those results are incorporated into test cases or used to improve a wide array of driver model behaviours [25]. Modelling this complexity even in state of the art simulations is difficult [26].

With the availability of big data, research has turned towards machine learning algorithms that can be used to infer a probability of future events derived from learned patterns. BT Morris et al. [27] describe several approaches for using machine learning algorithms for the domain of behaviour modelling. Long Short Term Memory (LSTM) has become popular to predict and generate temporal information [28].

IV. Methodology

A. Overview

Design Science is a research paradigm in which the goal is to solve identified problems through performing a set of activities with a resulting artefact. The Design Science framework is used to assess and refine an artefact through building and evaluating in an iterative process. It is accomplished by using and adding to the knowledge domain through rigour, and the application of a solution that reflects the business need [29].

Our artefact provides an approach on how to automatically generate collision paths using an RNN. We utilise TensorFlow (version 1.0) a machine learning library from Google and Python (version 3.4) to implement our model. Fig. 8 illustrates the key components of our design process. The key driver to undertake our research is the need to improve the test procedure of Active Safety Systems through the automation of scenario generation for testing. We add to the *Environment* shown in Fig. 8 a prototype application to demonstrate end-to-end accident scenario generation and to the *Knowledge Base* propose a set of metrics to evaluate the quality of generation. These metrics help us understand how accurately our RNN model is able to generate accident scenarios. A good generated accident scenario should be plausible to the given situation.

We utilise a Recurrent Neural Network (RNN) used in other domains where temporal awareness is important. Our RNN model will be used to generate scenarios which include the speed and direction of two colliding vehicles at each point in time and that are also reactive of independent actor/s, in our case traffic light states at an intersection that each vehicle approaches.

We start our solution by examining time series data prediction [3] and iteratively develop our prototype for multiple time series of multiple actors. We re-use helper functions[4] to separate data into training, validation and test sets that the model is fed. Data is separated into 80% training, 10%

[2]http://iglad.net

[3]https://github.com/pusj/LSTM-Time-Series-Analysis-using-Tensorflow

[4]https://github.com/pusj/LSTM-Time-Series-Analysis-using-Tensorflow

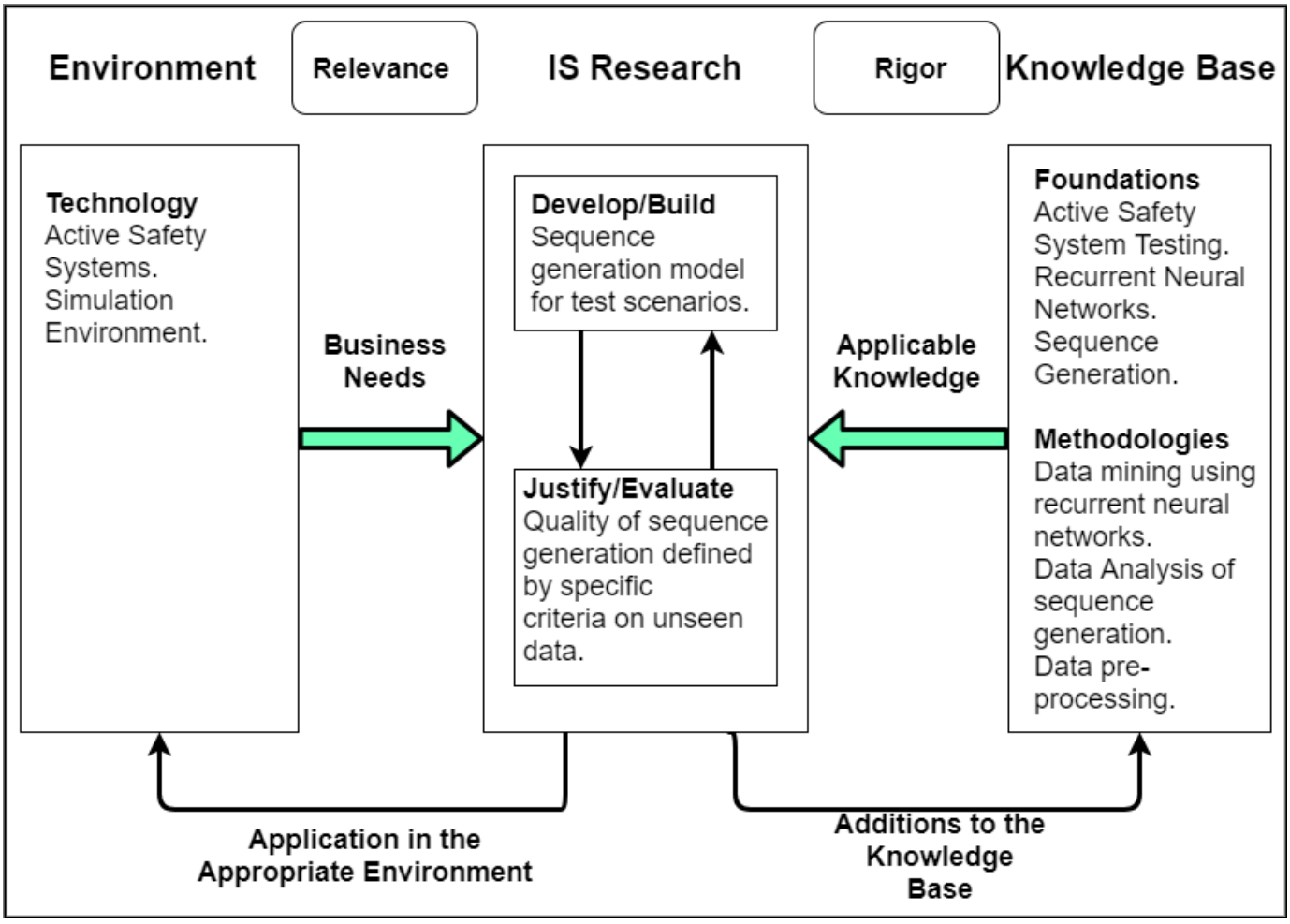

Fig. 8. Design Science Framework.

validation and 10% test sets. An example of the data format can be seen in Table I.

B. Artefact: Collision Model

Our artefact includes a design and implementation of a neural network model, that we call Collision Model, exemplified in Fig. 9. Input Features contain a subset of data shown in Table I and are fed into the Collsion Model in batches. Data distributed to separated neural networks and is not predetermined. The LSTM cells used are described by Zaremba et al. [30].We evaluate how data can be processed, which variables are important and the architecture of the model to achieve the best qualities of scenario generation. These include traffic light states, normalisation, scenario distinction, timesteps and a single versus parallel network:

1) : Light States during intersection Constraining a model to make it simpler and reduce the risk of overfitting is called regularisation. Traffic light data is collected as an array of length 3 eg; [0, 1, 1]. Each element is a binary value where 'go' = 1 or 'stop' = 0 and where the array represents [left, straight, right]. The traffic lights are limited to four states: 'all stop', 'go left', 'go right and straight' or 'all go'. The total value of each array is summed respectively to 1, 2, 3 or 4 in order to represent each state as a single integer. We added a 5th "null" traffic light state for all vehicles inside an intersection perimeter. This was done to measure whether traffic light states of a next intersection, which the cars record, will affect cars in its current intersection.

2) Normalisation: The entire dataset for each corresponding feature was normalised using feature scaling (12) where $x \in X$ and X is the input feature set. The features are normalised between a range of 0 and 1 before feeding them into the model for training, as normalising has been shown to help the rate of convergence [31]. Batch normalisation is suggested between input-hidden layers in a network, however we opted to normalise the entire data set prior to input as our data set was of a feasible size [32]. The input features of the model were normalised values of light, speed, direction.

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{12}$$

3) Scenario Distinction: A key is added at the end of each scenario to remove overlapping timesteps between scenarios. An alternative would have been to add zero values and use sequences of variable lengths. One downside to our method is that we lose some sequences by n-1 timesteps.

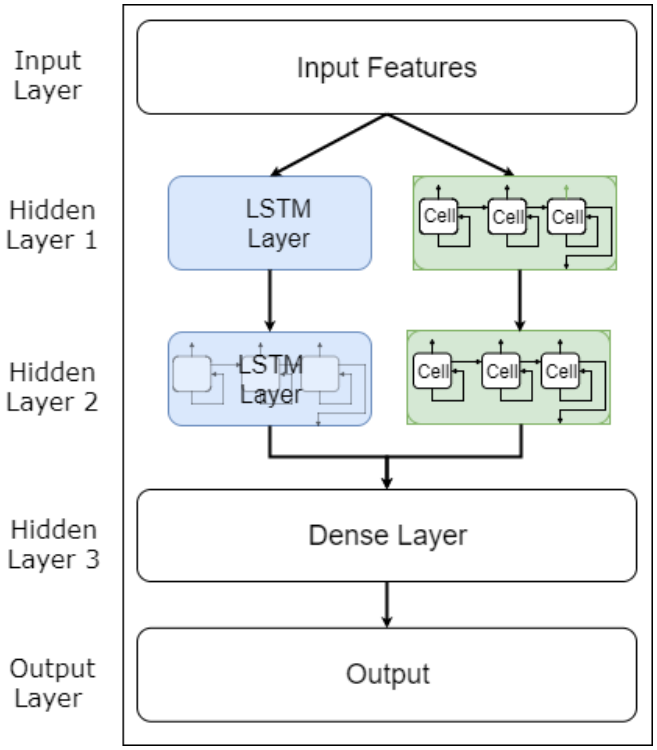

Fig. 9. Collision Model: Input Features consist a batch of pre-processed data collected from the simulator, it details the state in which vehicles and traffic lights are in. LSTM layers captures the learning process. The Output Layer gives a prediction of a next world state.

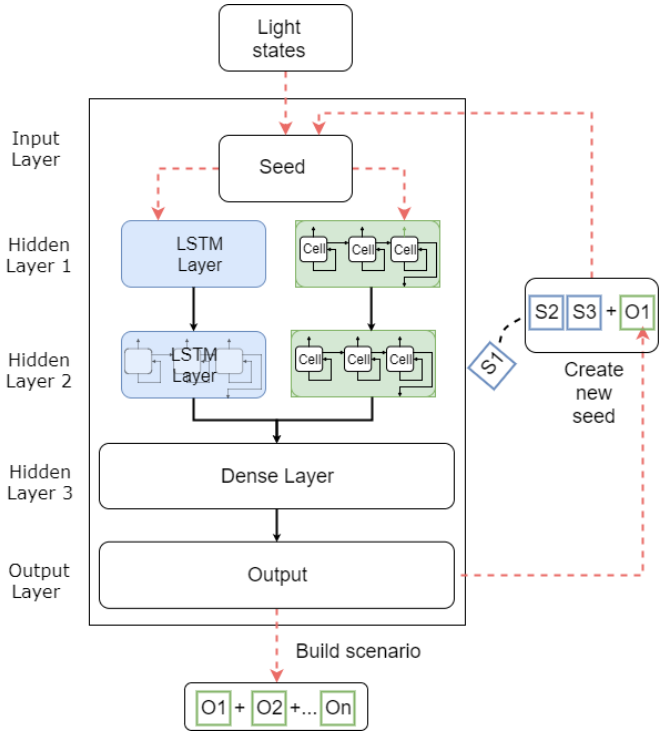

Fig. 10. Generation Model: The Input layer to the model is an initial seed: S1, S2 and S3 which are three sequential world states of light, speed, direction for vehicle A and light, speed, direction for vehicle B. Hidden Layers 1 and 2 are LSTM layers that process the input data. The Output Layer gives a prediction of a next world state. To build an entire scenario a new seed is made. Prediction O1 is concatenated to the S2 and S3 but the predicted light state is removed and an external light state is inject into the new world state.

4) Timesteps: We train the Collision Model using different length timesteps. Using one timestep would be equivalent to a standard forward-feed network. We model using 1-4 timesteps.

5) Single and Parallel networks: We model data using up to 192 neurons with a single network of two layers, 62 and 128 neurons and parallel network with 16 and 32 neurons respectively.

C. Aretfact: Scenario Generation Model

A second inclusion to the approach is Scenario Generation Model (GM) show in Fig. 10 where we describe how we generate an entire collision scenario.

D. Evaluation

Applying our artefact for test generation in Active Safety System development is likely novel and as such we will be providing a proof of concept. In the following paragraphs we describe how we will evaluate our artefact:

1) Analytical: We will analyse the feasibility of our approach in terms of what pre-processing is needed from raw data. As sensor data is available from in-vehicle systems, standards need to be established in the automotive domain for machine learning.

2) Descriptive: We will further determine the success of our approach through looking at the quality of scenarios generated in comparison with ground truth data. The quality is defined by the following metrics:

1) A collision occurs defined by two vehicles intersecting at the final point from generated paths.
2) The distance travelled from both vehicles.
3) The total speed at collision from both vehicles.
4) The cumulative angular change from both vehicles during a scenario.

E. Data collection

We source our data from the publicly available road traffic simulator[5] shown in Fig. 11 to train, validate and test our RNN implementation. The source code for the simulator can be found here[6]. The simulation is built using models such as the Intelligent Driver Model (IDM) and MOBIL [33] that define rules and behaviour of vehicles. As the simulator has no built-in data logging system we have had to implement this as well as a collision detection algorithm. The simulation runs at 60 frames per second and for each frame we can collect each individual vehicle position, state of upcoming traffic light, speed, direction and whether it has collided or not. There are also definable states concerning the environment including number of roads, intersections, traffic light switching interval and time-factor which is a coefficient that determines the rate at which the simulations runs.

[5]http://volkhin.com/RoadTrafficSimulator/
[6]https://github.com/volkhin/RoadTrafficSimulator

TABLE I
EXAMPLE OF RAW DATA REPRESENTING TWO VEHICLES INVOLVED IN A COLLISION SCENARIO

Car 1						**Car 2**					
ID	X	Y	TLS	Speed	Direction	ID	X	Y	TLS	Speed	Direction
333	-29.23042	-19.25	5	0	0	1233	-33.25000	-11.27591	3	0.2333883	-1.570796

Each scene contains 12 data-points as seen in Table I which include the car identification number, x-coordinate, y-coordinate, traffic light state (TLS), speed and direction for the two cars involved in the collision.

After running the simulation with various map configurations, different numbers of cars and different time-factors, we concluded that for the purposes of our research we could run the simulation in the configuration seen in Fig. 11 with 100 cars at a time-factor of 2. The total amount of frames collected was determined by logging high speed vehicles from entry into the lane boundary highlighted in green up until collision in the red intersection which was approximately 50 frames.

We focused our data collection around a signalised intersection highlighted in red in Fig. 11 due to the complex and possible interesting scenarios that appear there. 100 cars are added to the simulation in designated areas highlighted in blue in order to avoid creating immediate collisions. Data logging begins whenever a car enters one of the roads highlighted in green connected to the intersection. If a collision occurs within the red area then data of the past 50 frames of the two vehicles involved are saved as a scenario, the two vehicles are subsequently removed from the simulation and two new cars are added in the designated blue areas.

With this configuration of the simulator we were able to record 4835 collision scenarios containing 241750 scenes with an estimated total running-time of a week. The simulator was monitored as there were instances where it would become congested by the vehicles and the simulation world had to be reinitialised. The reasoning for a time-factor of 2 was due to the need of finer data for the RNN. If a vehicle drove down one of the roads towards the intersection at a very high speed then we observed that a time-factor greater than 2 would result in scenarios containing a lot less scenes as the logging frequency is static.

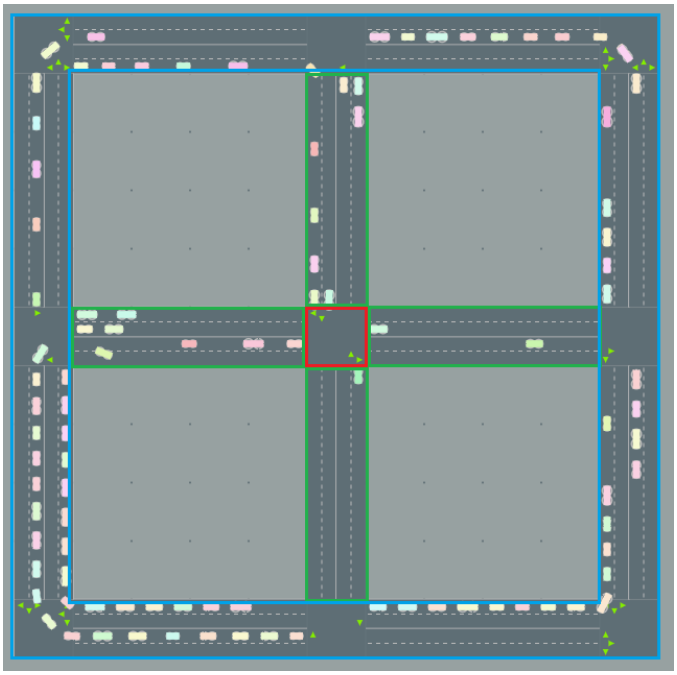

Fig. 11. Road traffic simulator

V. RESULTS

In this section we describe the quality of generated scenarios in respect to our evaluation criteria in the subsection: Generated Collision Scenarios, and the development of our model in the subsection: Training. Independent variables experimented with in Training are timesteps, normalised variables, light states, network shape where the dependant variable is the Mean Square Error.

A. Generated Collision scenarios

The model used to generate collision scenarios consisted of a parallel network with 16 and 32 neurons respectively. Data was regularised, normalised using feature scaling, had a learning rate of 0.01, and trained for 200 epochs. Total amount of data used for the model consisted of approximately 120875 scenes (2417 scenarios). We used 237 seeds from the test dataset, consisting of 3 timesteps each, to generate scenarios that are 47 timesteps long.

From the 237 seeds, the model was able to generate 32 (13.5%) successful collision scenarios (SCS) and 205 (86.5%) unsuccessful collision scenarios (UCS) according to criteria 1. Fig. 12 shows where the collisions from the training set (green dots), test set (blue squares) and the successfully generated occurred (red stars). Fig. 13, 14 and 15 show the generated scenario in red that were created using the seed from the corresponding ground truth scenario indicated in blue. These figures show the trajectories (dotted line), the final positions (rectangle) and directions (arrow) of vehicles. Fig. 13 shows the most common SCS 46.88% where two vehicles are moving with similar speeds. Fig. 14 shows the second most common SCS 37.5% where one vehicle is moving and the other is stationary. Fig. 15 shows the third most common SCS 15.63% where one vehicle is moving at a higher speed than the other. Fig. 16 is an example of a UCS where one vehicle has veered off path and travelled an insufficient distance. In order to present our results for criteria 2, 3 and 4 we use Kernel Density Estimation (KDE), which is a non-parametric technique used for visualising the underlying distribution of our data [34]. For each KDE plot (Fig. 17, 18 and 19) we include the distribution of the whole training set (purple), all of the ground-truth data used for SCS (blue), all of the UCS data (green) and all of the

Fig. 12. Collision points at intersection

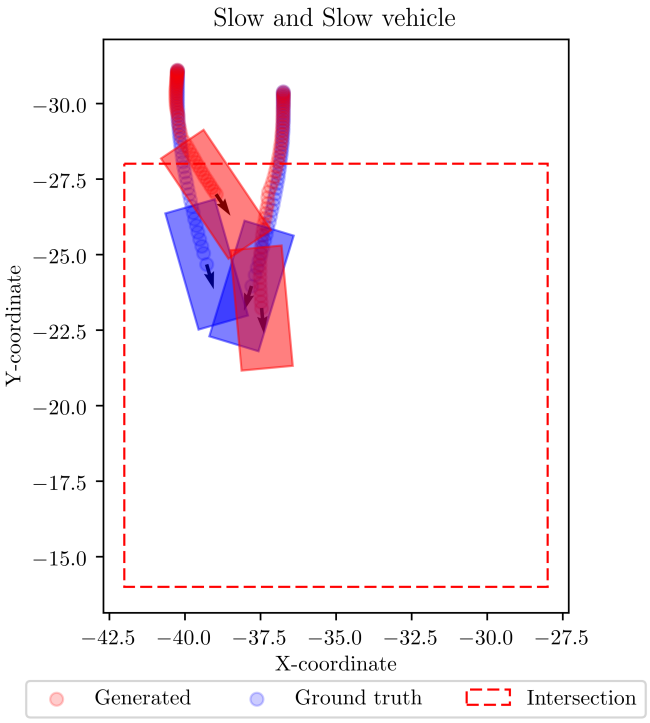

Fig. 13. Most common SCS

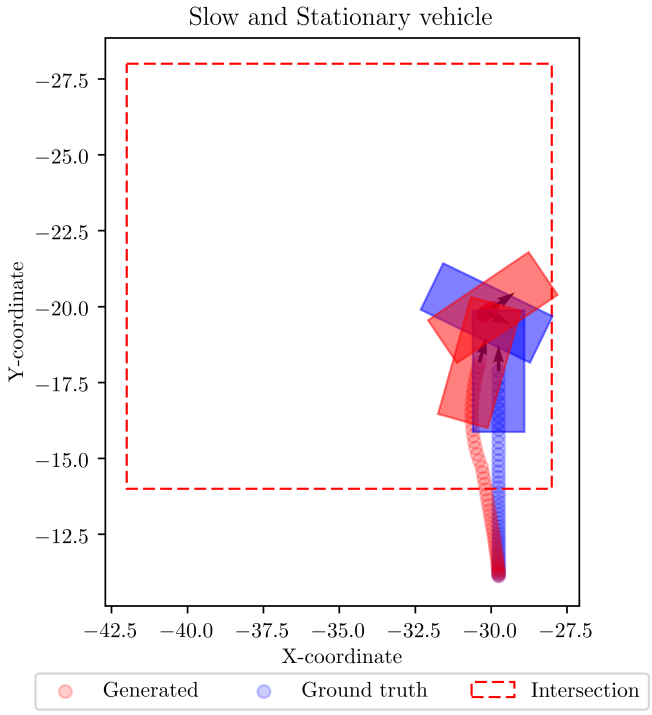

Fig. 14. Second most common SCS

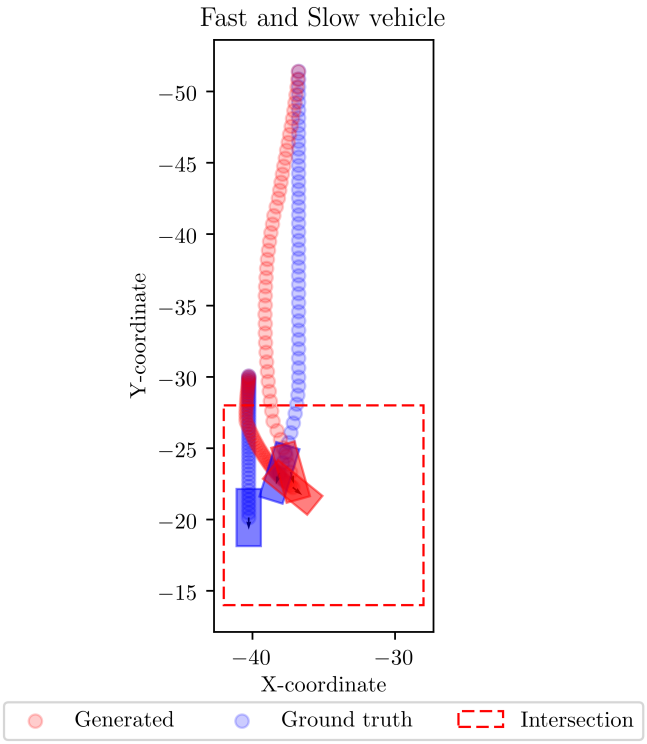

Fig. 15. Third most common SCS

SCS data (red). Fig. 17 shows the distribution of the distance travelled by both vehicles combined during a scenario, Fig. 18 shows the distribution of the combined speed of both vehicles at the point of collision and Fig. 19 shows the cumulative angular change in radians by both vehicles combined during a scenario. From Fig. 17 we see that distances generated from SCS have a similar distribution to the ground truth values. The total speed at collision shown in Fig. 18 is skewed left towards the lower values of speed as is the same with ground-truth data however with a much greater density more in line with that of the training data. The model however fails at

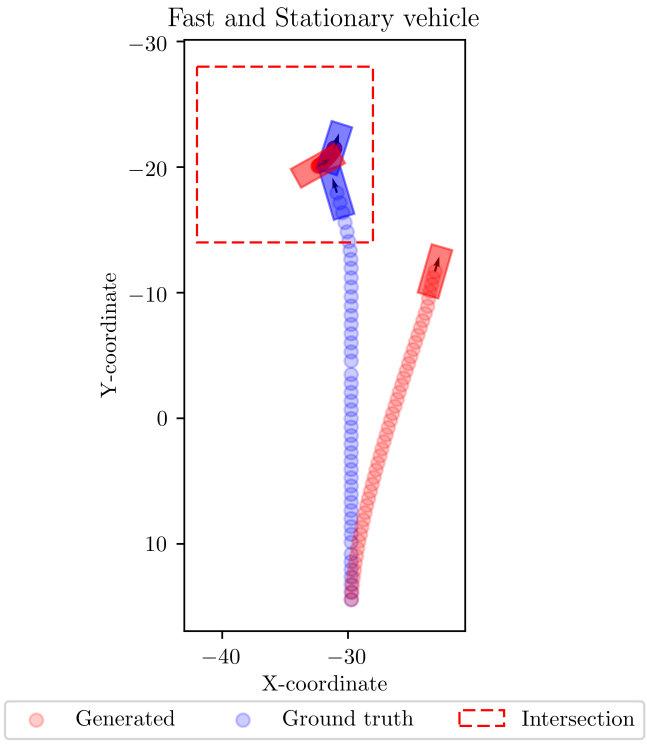

Fig. 16. Example of a UCS

properly capturing the change in direction seen in Fig. 19. The training data and ground truth data are heavily skewed to the left meaning that the vehicles barely change direction during a scenario whilst the generated scenarios are skewed to the right leading to greater changes in direction.

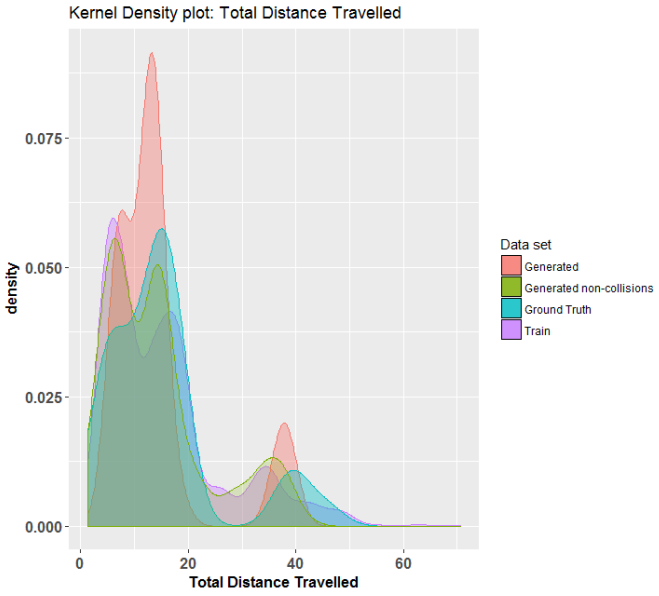

Fig. 17. Evaluation criteria 2

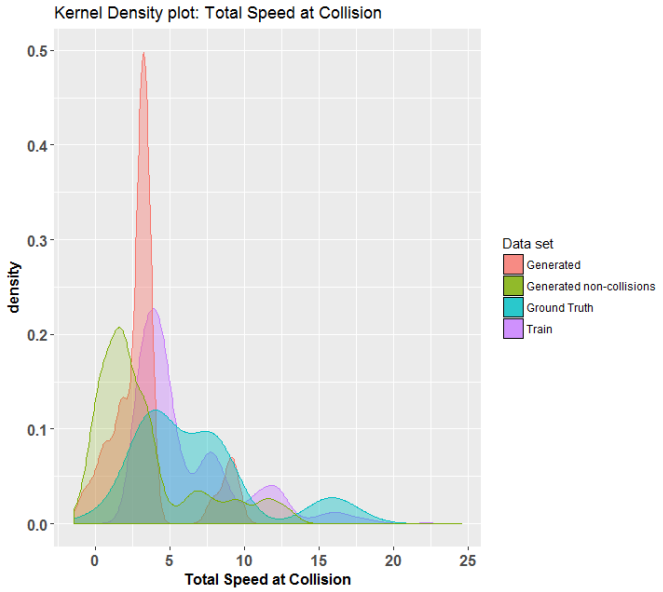

Fig. 18. Evaluation criteria 3

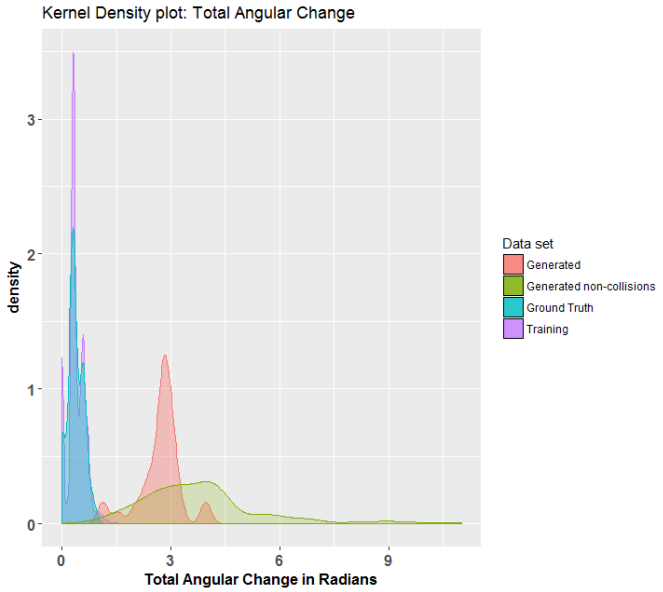

Fig. 19. Evaluation criteria 4

B. Training

1) Timesteps: We saw that using more than 1 timestep significantly improved the error by 55.24%. Table 2 shows the errors produced between 1-4 timesteps. The lowest error recorded was produced at 3 timesteps at 13.54% lower than the nearest value at 4 timesteps. A use of more than 3 timesteps saw a rise in error. Up to 10 timesteps were tested under slightly varying parameters (omitted due to the potential of discrepancy).

2) Light States: We also tested the effect of having light states for vehicles once they travel inside the intersection boundary shown in Fig. 11. Removing light states inside the intersection shows a 5.7% decrease in error.

3) Single and Parallel networks: Halving the neurons per layer and then adding a second network, so that we have two networks of 16 and 32 neurons for each layer, improves how the network was able to converge. A single network with equal neurons showed that after 200 epochs the error was of near equal values seen in Table II, but did not converge as well as seen in Fig. 20.

TABLE II
TRAINING MSE VALUES

Timesteps	1	2	3	4
MSE	3.7181	2.0425	**1.6643**	1.9249

Light State	Original	Null-state
MSE	0.00742	**0.0066**

Network	Single	Parallel
MSE	**0.0021**	0.002

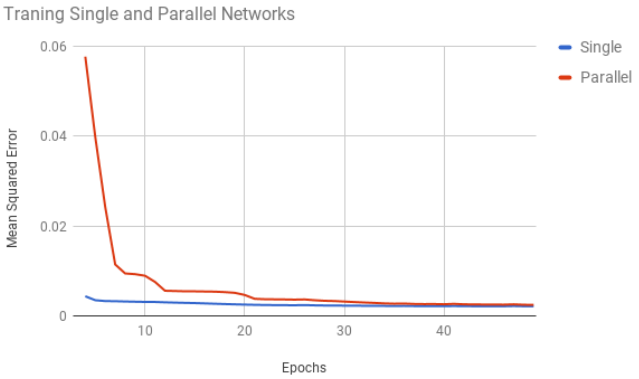

Fig. 20. Parallel vs Single network with equal amount of neurons.

VI. DISCUSSION

A small yield of successfully generated collision scenarios were produced over seeds that had a distribution close to population of the data set. It is important to notice that having this comparable distribution yields towards the scale of a critical test. We would expect to see an increase in the number of collisions if we aimed to generate within certain parameters. Increasing the number of layers or neurons did not yield better results beyond our model. Deep layers offer better feature extraction but are difficult to penetrate while too many neurons reduce a training loss but lead to overfitting so often multiple training strategies need to be considered [35].

Evaluating the density plots we notice that the cumulative angular change distribution is skewed to the right compared to the ground-truth and training data. It is clear that angular data had not been accurately modelled. We reason that the direction a vehicle will travel is dependent on the context of its position, X and Y coordinates in our dataset. This was omitted during training since excluding the X and Y coordinates as input features generated a lower loss (MSE) value. Whilst training, we use the MSE as an indication of how well the model was converging. One possible solution for improving the directional modelling could be to use a value that represents the change in direction for each vehicle rather than the true direction of the vehicle with respect to the simulation environment.

Understanding how to represent and model data came through iterative process with the results motivating further progress. For instance granularity of the data needed to be considered as well as how far back in time data is collected is relevant. We were able to model scenarios, generating 47 timesteps in the future using 3 timesteps of starting sequential data. We did not test variable time lengths for slow and fast moving vehicle but should be considered for future work.

The notion to remove light states during an intersection was an intuitive choice from examining the behaviour of the simulator and real world driving. We felt it appropriate to measure the effect of removing light states which showed a decrease in error. This could suggest that vehicles inside an intersection are not reacting to lights states, at that time. Simply identifying an independent variable is not enough in this case, so consideration should be taken into place that available data from in-vehicle systems used for modelling may have a conditional influence.

Dropout is a strategy to improve generalisation against overfitting but RNNs find it hard to benefit from it when used between neurons because important temporal information is lost. Zaremba et al. [30] demonstrated that better results can be obtained when dropout is applied between layers and not between neurons. We experimented with a different approach, splitting a network into two separated networks to force constraints on learning. We did not discriminate which features should be modelled in each network. The rate of convergence was higher and the error after training was near identical for the single network. This suggests that the parallel model was better able to generalise.

A. Alternative Approaches

Our chosen approach is an initial step towards the automatic generation of collision scenarios using an RNN for active safety system testing, which is reflected in our process and complexity of the models we built. There exist more complex approaches that tackle similar problems of modelling trajectories and sequence generation but the implementation of those were deemed out of scope for this research paper.

A. Alexandre. et al [36] propose, what they call, a ”Social-LSTM” model for the task of predicting the trajectory of multiple humans in a scene. Their model uses a LSTM network for each individual trajectory and, depending on the spatial proximity of the individuals, pool these networks together in order to allow the LSTM networks to share information. This is done as individuals will alter their paths depending on the motion of their neighbours who in turn do the same. Their method outperforms other state-of-the-art methods. Implementing a similar approach could increase the accuracy of prediction for our collision model.

Another viable approach would be to use the framework of a Generative Adversarial Network (GAN) for sequence generation [37]. The framework consists of training a generative

model and discriminate model at the same time, then pitting them against each other in a form of unsupervised learning. To put it into context we can generate an entire scenario and use a discriminative model to determine whether or not the collision scenarios are real. Both models compete to improve their performance until generated scenarios are indistinguishable from the real scenarios.

B. Threats to validity

As our approach is a proof of concept and to the extent of our own research in this domain, also novel, it is not validated against other approaches. It is not implemented using real life data nor evaluated in a real testing environment used in the automotive industry. Data used from the simulation is a simplified example of patterns that may occur at a real-world intersection. We also assume that real world intersections have some pattern in which vehicles react to actors in an environment.

The metrics chosen to evaluate the generated scenarios were defined by the researchers themselves as deemed appropriate. For criteria 1 the researchers rely on human observation whether or not the two vehicles intersect by looking at each generated graph. The size of the vehicles in the simulation environment vary between 3-5 units in length but with a static width of 1.7 units. This was not recorded at time of data collection so a length of 5 units was given to each vehicle in the generated scenarios. When calculating the X and Y coordinates of the generated scenarios a time constant had to be derived in order to calculate the next coordinate as the speed value defined in the simulation environment is arbitrary. The time constant has been validated against ground-truth data in order to mitigate this but still leads to some uncertainty. The use of techniques such as Dynamic Time Warping [38] in order to analyse the similarity between trajectories were considered but due to time constraints were never used.

We learned that MSE values were not always a precise indicator of good generation. Generation could have been done earlier in our process to verify that any changes were a good choice. Our model does generate scenarios but there still remains the problem of automatically assessing the quality of the generated scenarios without having to observe the generated paths and analysing statistics.

VII. Conclusion and future work

We use an RNN to generate collision scenarios and the quality of those generations are measured by a set of metrics. A key driver of our paper was the need to find a solution with reasonable effort to accurately automate generation of scenarios for Active Safety testing and crucially for the advent of Autonomous Drive. Our results show that an automated generation of a collision scenario is by some means achievable using a recurrent neural network. We acknowledge that with additional data mining techniques and a complex model our results may have been improved. The scope of our thesis was to provide an approach and we reason that while RNNs have been applied in other domains to related problems where temporal data is important, approaches in general for automating the generation of collision scenarios are sparse. Whilst the percentage of collisions achieved from a test set with a distribution comparable to the population is relatively low, the qualities that have been derived from successful collisions and non-collisions alike are interesting enough in relation to distance, speed at impact. Future work in this domain would include a thorough assessment of the features available and needed as well as implementing our approach with the use of real-world data. Our results pose questions towards experimenting with other accident data sets and combining RNN with other artificial intelligence methods.

Acknowledgment

We would like to thank Alessia Knauss of Chalmers University of Technology and Hang Yin of Chalmers University of Technology for their guidance and invaluable feedback.

References

[1] Z. Sun and S.-K. Chen, "Automotive active safety systems [introduction to the special section]," *IEEE Control Systems*, vol. 30, no. 4, pp. 36–37, 2010.

[2] "GIDAS," http://gidas.org/en/willkommen/, accessed: 2017-10-23.

[3] 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.

[4] P. Zhang, "Zhang, g.p.: Time series forecasting using a hybrid arima and neural network model. neurocomputing 50, 159-175," vol. 50, pp. 159–175, 01 2003.

[5] 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, year = 2014, url = http://dx.doi.org/10.1155/2014/614342, doi = 10.1155/2014/614342, biburl = http://files.hindawi.com/journals/jam/2014/614342.enw, articalid = 614342, pages = 7.

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

[7] A. Knauss, J. Schroeder, C. Berger, and H. Eriksson, "Paving the roadway for safety of automated vehicles: An empirical study on testing challenges," in *IEEE Intelligent Vehicles Symposium, IV 2017, Los Angeles, CA, USA, June 11-14, 2017*, 2017, pp. 1873–1880. [Online]. Available: https://doi.org/10.1109/IVS.2017.7995978

[8] T. M. Mitchell, "Machine learning. 1997," *Burr Ridge, IL: McGraw Hill*, vol. 45, no. 37, pp. 870–877, 1997.

[9] A. Graves, *Supervised Sequence Labelling with Recurrent Neural Networks*, ser. Studies in Computational Intelligence. Springer, 2012, vol. 385. [Online]. Available: https://doi.org/10.1007/978-3-642-24797-2

[10] 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.

[11] 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.

[12] A. Graves *et al.*, *Supervised sequence labelling with recurrent neural networks*. Springer, vol. 385.

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

[14] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer, "Defining and substantiating the terms scene, situation, and scenario for automated driving," in *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*. IEEE, 2015, pp. 982–988.

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

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

[17] M. Benmimoun, F. Fahrenkrog, A. Zlocki, and L. Eckstein, "Incident detection based on vehicle can-data within the large scale field operational test "eurofot"," in *22nd Enhanced Safety of Vehicles Conference (ESV 2011), Washington, DC/USA*, 2011.

[18] 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.

[19] A. Knauss, J. Schroeder, C. Berger, and H. Eriksson, "Paving the roadway for safety of automated vehicles: An empirical study on testing challenges," *NA*, 2017.

[20] A. Jansson, E. Olsson, J. Linder, and M. Hjort, "Developing of a driver model for vehicle testing," in *14th International Symposium on Advanced Vehicle Control (AVEC), Tokyo, September 2014.*, 2014.

[21] S. S. Mahmud, L. Ferreira, M. S. Hoque, and A. Tavassoli, "Application of proximal surrogate indicators for safety evaluation: A review of recent developments and research needs," *IATSS Research*, 2017.

[22] 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.

[23] 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.

[24] Y. Matsui, S. Oikawa, and M. Hitosugi, "Analysis of car-to-bicycle approach patterns for developing active safety devices," *Traffic injury prevention*, vol. 17, no. 4, pp. 434–439, 2016.

[25] N. AbuAli and H. Abou-zeid, "Driver behavior modeling: Developments and future directions," *International Journal of Vehicular Technology*, vol. 2016, 2016.

[26] F. Huang, P. Liu, H. Yu, and W. Wang, "Identifying if vissim simulation model and ssam provide reasonable estimates for field measured traffic conflicts at signalized intersections," *Accident Analysis & Prevention*, vol. 50, pp. 1014–1024, 2013.

[27] B. T. Morris and M. M. Trivedi, "Understanding vehicular traffic behavior from video: a survey of unsupervised approaches," *Journal of Electronic Imaging*, vol. 22, no. 4, pp. 041 113–041 113, 2013.

[28] A. Karpathy, J. Johnson, and L. Fei-Fei, "Visualizing and understanding recurrent networks," *arXiv preprint arXiv:1506.02078*, 2015.

[29] R. H. Von Alan, S. T. March, J. Park, and S. Ram, "Design science in information systems research," *MIS quarterly*, vol. 28, no. 1, pp. 75–105, 2004.

[30] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[31] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient backprop," in *Neural networks: Tricks of the trade*. Springer, 2012, pp. 9–48.

[32] C. Laurent, G. Pereyra, P. Brakel, Y. Zhang, and Y. Bengio, "Batch normalized recurrent neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*. IEEE, 2016, pp. 2657–2661.

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

[34] Y.-C. Chen, "A tutorial on kernel density estimation and recent advances," *arXiv preprint arXiv:1704.03924*, 2017.

[35] D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent, "The difficulty of training deep architectures and the effect of unsupervised pre-training," in *Artificial Intelligence and Statistics*, 2009, pp. 153–160.

[36] 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.

[37] 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.

[38] C. A. Ratanamahatana and E. Keogh, "Everything you know about dynamic time warping is wrong," in *Third Workshop on Mining Temporal and Sequential Data*. Citeseer, 2004, pp. 22–25.