Leveraging Vehicular Communications in Automatic VRUs Accidents Detection

Bruno Ribeiro, Maria João Nicolau and Alexandre Santos
Algoritmi Center, Department of Informatics,
University of Minho, Campus de Gualtar,
4710-057 Braga, Portugal
bruno.ribeiro@di.uminho.pt, joao@dsi.uminho.pt, alex@di.uminho.pt

***Abstract*—As technology advances on the field of Vehicular Ad hoc Networks (VANETs), there is a growing concern within the research community regarding the safety of the the Vulnerable Road Users (VRUs). These entities play an important role in traffic, but their typical agility and difficult to predict behavior pose challenges in the development of automatic systems that aim to protect them. The application of Machine Learning (ML) techniques on top of the communication data that can be collected from the road environment has the potential to predict VRUs movement, detect/locate them, or even compute probabilities of collisions. This paper proposes an automated and real-time VRU accident detection system (focused on motorcycles) by using neuronal networks with communication data that is generated by means of simulation, using the VEINS framework (coupling *SUMO* and *ns-3*). Results show that the proposed system is able to automatically detect any accidents between passenger vehicles and motorcycles at an intersection within 1 second, with an average of 0.61 second, after its occurrence.**

I. Introduction

Intelligent Transportation Systems (ITS) are systems that consist on an intricate set of technologies applied to road agents (e.g. vehicles, pedestrians, infrastructures) that aim to provide a more efficient and safe usage of the roads - allowing, for example, to control traffic operations or influence drivers behavior. These systems enable the implementation of several applications that, relying on information that is exchanged between the road agents, allows entities (e.g. drivers) to make smarter choices - either manually or automatically. These applications can range from simple day-1 use cases (e.g. *Emergency Vehicle Warnings*) to more advanced solutions, such as *advanced automatic accident detection*.

Road agents equipped with communication capabilities are able to exchange important knowledge that can help saving lives by minimizing the effects of accidents, improve traffic flow, and so on. Naturally, given the increasing number of equipped devices on ITS environments that are exchanging such information (e.g. information about the vehicles, the infrastructures, the road/traffic conditions), there is an huge amount of data that is generated with high frequency. In this context, it is pertinent to analyze the feasibility of using this large volume of data to implement automatic processes that allow actions such as predicting traffic jams, calculating alternative routes, computing probabilities of accidents, etc.

Thus, applying ML techniques on said communication data has the potential to improve traffic flow and mobility, and also improve road safety. Improving road safety is particularly important for Vulnerable Road Users (VRUs), since they are the most exposed road agents and have a high casualty rate. Typically, these users usually do not have a protective external shell and possess poorer safety mechanisms when compared to normal passenger cars or trucks. Due to their nature of being highly mobile and move in a way that tends to be difficult to predict by traditional methods, ML techniques may be used to implement more advanced systems that try to predict such actions and prevent accidents, or minimize their effects.

This implementation is possible if these users possess communication capabilities that allow communication between themselves and other road agents. Although entities such as bicycles and pedestrians may possess communication capabilities (for instance, using smartphones), it is unlikely that they may communicate directly with other vehicles on the road - which are typically equipped with IEEE 802.11p transmission capabilities. On the other hand, motorcycles are fairly easy to equip with On Board Units (OBUs) that possess communication technologies similar to regular vehicles. Hence, from the VRUs group, motorcycles make the better subject to study the potential use of automatic solutions for accident detection, which may *passively* improve their safety - the detection may, for instance, trigger emergency services that may reduce the severity of injuring after accidents or even save lives. Even if motorcycles and regular passenger vehicles are not equipped with sophisticated safety systems that enable them to detect that they are involved in an accident, it is possible to resort to Road Side Units (RSUs) (infrastructure units on road environments) to implement such advanced safety systems. Naturally, there are other kind of safety methods that may be also able to predict and avoid accidents, resorting to sensors or advanced mechanisms (e.g. cameras, RADAR, LIDAR) that allow to obtain environment knowledge and activate active safety measures, such as emergency breaking, automatic steering or airbag deployment. However, this type of solutions may perform poorly in situations where line of sight is non-existent or limited (e.g. the VRU is in a blind spot, behind a parked vehicle). This situation is aggravated when considering VRUs, for their smaller size and high mobility, which make them harder to be detected. Hence, the implementation of such systems resorting to wireless communications between VRUs (motorcycles in particular) and regular vehicles/infrastructure

may have a great impact on the general safety of road agents. This work consists in the development and test of a VRUs (motorcycles) accident detection system, resorting to Machine Learning (ML) techniques. The system consists of two essential pieces - the simulation scenario and the ML models.
This paper is organized as follows. Section two presents the state of the art. Section three describes the development of the simulation scenario. Section four describes the process of building the ML models and how the data was collected and treated. Section five discusses the main results of this work. Finally, section six reviews the main conclusions.

II. Related Work

VRUs play a very important role on traffic flow. Most VRUs are typically very agile (e.g. pedestrians, motor-cyclists) and their movement/behavior is hard to anticipate (sometimes not even in compliance with traffic rules). Hence, detecting or predicting their behavior is a difficult task. However, the application of machine learning techniques on vehicular environments data has the potential to detect or predict VRUs movement, classify their behavior and even compute probabilities of collision with them. These solutions may enable the avoidance of accidents and thus also achieve a more efficient and safer traffic flow.
Most of the relevant related works found tend to focus on the prediction of the movement/intentions of VRUs or post-accident analysis, and not on the detection of the incident itself. Furthermore, the data that is analyzed in such works tends to be collected by means of sensors/camera-like systems and not on active communication systems. This section discusses some interesting related works that focus on machine learning methods applied for the safety of VRUs.
In [1] a set of VRUs movement models based on machine learning techniques are present, aiming to classify VRUs current motion state and to predict the upcoming trajectory. The *dataset* consists of over a thousand pedestrian and near five hundred cyclist scenes acquired at an urban intersection. The data was collected using a mix of cameras and laser scanners. The recognition of the motion state and the trajectory prediction is then tested using a method of polynomial approximation. The results show higher classification values and the system is able to recognize motion state changes earlier, compared to Interacting Multiple Model (IMM) *Kalman Filter*.
In [2], a VRU trajectory prediction service is presented, using regression algorithms on Cartesian coordinates data. Using Alternating Model Tree (AMT), the next position is predicted with an error of less than 3.2 centimeters, increasing up to 1 meter when predicting the next 5 positions (1s between consecutive positions). As future works, the authors plan to use this service to estimate the collision probability.
In [3], the authors compare the use of different machine learning algorithms in the identification of crash severity factors of different Vulnerable Road User Groups (pedestrian, bicyclist and motorcyclist), using real data from Queensland, Australia (from 2013 to 2019). Random Forest classification models performed more robustly in test accuracy: (motor-cyclist: 72.30%, bicyclist: 64.45%, pedestrian: 67.23% and unified VRU: 68.57%).
In [4] introduces a model that identifies risk factors for VRUs that can affect their injury severity when involved in an accident. To train the model, records from VRUs related crash data were analyzed. The results indicated that Decision Tree (DT) outperformed Logistic Regression (LR), since the coefficient correlations were not considered and all the variables were taken into account - the model revealed to be more accurate considering the crash severity data under evaluation. Nevertheless, both methods could correctly classify the classes with relatively high accuracy.
The work in [5] analyses injury severity of three-wheeled motorized rickshaws, resorting to several algorithms - Decision Jungle, Random Forest, and Decision Tree - and data from the city of Rawalpindi, Pakistan. Results showed that Decision Jungle outperformed the other solutions with an overall accuracy of 83.7%. The analysis also showed that features such as the lighting condition, younger drivers, high-speed facilities, weekdays, off-peak, and shiny weather conditions were more likely to worsen injury severity of the crashes.

III. Simulation Scenario

Until this point, it was not found on the literature any datasets that contain VRUs related accidents using information that was gathered from ITS messages - hence the need to build datasets from scratch.
The development of advanced ITS systems requires a proper evaluation of its performance. However, performing field tests in vehicular environments is very challenging: the large number of nodes and traffic scenarios makes it very difficult to collect data in real experiments and developing real prototypes is a very expensive task, both in terms of money and time. On the other hand, the use of simulators is a very popular choice on the research community when it comes to the analysis of communication and transportation solutions, for its ability to perform assessments in a large scale.
In order to perform a proper simulation of vehicular environments, both a traffic simulator and a network simulator are required. For this reason, researchers tend to use frameworks that couple these two kind of simulators - some examples of these solutions are *VEINS, Artery* and *Eclipse Mosaic* (formerly known as *VSimRTI*). These frameworks couple the simulators in a transparent way for the user, facilitating the development of this kind of applications but still leaving space for refined parameterization, on both the communications and the mobility simulators.
In this work, the *VEINS* framework was chosen - since it was already used before on other related work, which facilitated the implementation of the new scenario.
The established scenario was inspired on ETSI standard use cases, namely *Collision Risk Warning from RSU* [6], *Motorcycle Approach Warning* [7] and *Turning collision risk warning* [8]. The scenario simulates collisions between passenger vehicles that are turning left on an intersection (red) and motorcycles that are following on the road (yellow), as

Figure 1 shows.

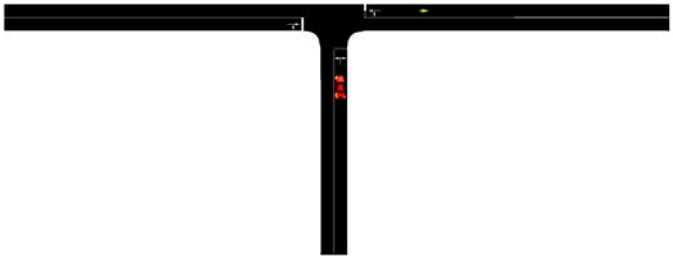

Fig. 1. *SUMO* scenario - Intersection Collisions

In order to simulate the effects of the accident itself, the vehicles involved on the collision halt on the lane for five minutes. The remaining vehicles proceed normally according to their movement model - which ultimately results in traffic jams during the accidents. After the accident, the vehicles then proceed normally on their predefined routes.

In terms of communication, all nodes on the simulation (passenger vehicles, motorcycles and also the RSU that is placed on the intersection) are equipped with the *IEEE 802.11p* technology to communicate - using the default value parameters on the configuration files (*txPower = 20mW; bitrate = 6Mbps; minPowerLevel = -110dBm*).

The vehicles (both passenger and motorcycles) are exchanging Basic Safety Message (BSM)-like beacons [9] with a 10Hz rate - the messages content is filled in VEINS with information that is extracted from the traffic simulator SUMO via the TraCI API. The beacons were defined to contain all the standard information that was possible to obtain through the simulation: *Station ID, Position (Longitude, Latitude, Elevation/Altitude), Heading, Speed, Acceleration, Vehicle Size (Length and Width), Vehicle Type* and a *Timestamp*.

The RSU is capable of receiving the beacons generated by the vehicles, but it does not send any messages throughout the simulation.

The simulation runs for a total of 24h. A total of ten simulations were performed, with different simulation seeds, generating ten different sets of data. Six of those datasets were later used to train the model, two for validation and the remaining two were used for testing.

IV. Accident Detection with machine learning

In this use case, we are trying to define, train and test a model that aims to detect if an accident is occurring between passenger vehicles and motorcycles on an intersection. To achieve so, two types of *neural networks* were tested: Multilayer Perceptrons (MLPs) and Long Short-Term Memory Networks (LSTM).

MLPs are a classical type of neural networks [10] and they typically consists of one or more layers of neurons. On this type of networks, data is fed into the input layer, one or more hidden layers provide levels of abstraction and predictions are made on the output layer. MLPs use a series of equations with inputs, outputs and weights, and transform inputs into singular outputs between 0 and 1. That generated output serves as input to another layer, and the process continues until a singular output is reached. In other words, it is a feed-forward neuronal network (the information moves only in one direction - from input nodes, through hidden nodes into output nodes). MLP was selected because it is very suitable for tabular datasets and in the classification prediction problems where inputs are assigned a label (on this case, *accident* or *not in accident*).

LSTMs [11] are a specialized type of Recurrent Neural Networks (RNN) architecture, capable of better learning long-term dependencies - the main advantage of LSTMs, when comparing to traditional RNNs, is that they retain information for longer periods of time, which in allows the early learned important information to also be impactful on the decision of the model, even if it is at the end of the sequence. LSTMs contain three internal layers acting on the state and input. These internal gates are the key to LSTM cells - they are weighted functions that govern the information flow (information state). The *Forget Gate* decides what information to discard from the internal state; the *Input Gate* decides which values from the input to add to the internal state; the *Output Gate* decides what to output based on the input and internal state (which information gets passed to the next state). LSTM was selected for its capabilities to process, predict and perform classification based on *time series* data.

In this use case, both models possess an input layer which expects (at most) 8 features - *Number of Vehicles, X Position, Y Position, Speed, Heading, Acceleration, Length and Width* and an output layer with a *sigmoid* activation function - which outputs a value in the range 0 to 1. Several variations regarding the number of hidden layers (and dropout layers) and set of input features were tested on each model. The best performing models are presented in the results section.

The remaining of this section describes how the data was gathered and processed, and also how the model was trained and tested.

The datasets that are built to feed the ML models consist of the collected messages that both passenger and motorcycle vehicles are exchanging through the use of communications. In order to build the data, the RSU that is installed in the intersection collects and saves all the received messages in that area to a Comma-separated values (CSV) file, line by line (each line corresponds to a collected message).

The ML model must be constantly updated with all the changes in the whole environment so that it can take them all into account when performing the classification. So, in order to overcome the problem of having a large collection of *singular* vehicular data, the second step taken was to aggregate data in a temporal fashion (turning individual records into environmental information): the dataset was split in fixed time intervals (1s, 0.5s and 0.1s were tested) and several methods were tested for aggregation (min, max, sum and average) - e.g. every message that was sent within one second, is now condensed into a single record. However, a new *Vehicle Count* feature is added to each record, in order to complement the aggregation data - which states how many different vehicles

sent messages during that period of time. Additionally, the *Station ID, Vehicle Type and Timestamp* features are now removed from each record, since they no longer make sense as aggregated information. The *Position Z (elevation)* feature is also removed - the simulation scenario does not consider this information - all values are equal to zero, making it irrelevant.
The main challenge in the defined use case (detecting accidents related to VRUs on a crossing) is that the dataset can be considered as unbalanced. There are only a few accidents in each simulation, hence only a few records on the dataset set to be *inAccident = true*. In a way, these records represent anomalies in the complete data (may be considered outliers). Thus, different class weights were estimated - the model's loss function is assigned higher value to the positive instances, which are rarer.
Several parameters were experimented when training and testing the models, in order to find which one performed better: aggregation time (1s, 0.5s, 0.1s); aggregation type (max, min, sum, average); number of neurons on the model's layers (32, 64, 128); different sets of features to feed the input layer of the models; and, in the LSTMs case - 5, 10, 15 and 20 timesteps were also tested.
A different model was trained and tested for each set of parameters, saving the results onto CSV files, using the format *[Real Value, Model's Prediction Value]* to allow a later more in-depth analysis.

V. Results

On this particular use case, simply analyzing the accuracy of the models to determine its performance is misleading. Although the high value of the metric may sound promising, the model's performance in terms of detecting the accidents may actually be poor. Since there are only a few accidents happening during the simulation time (they are rare events), only a few records on the dataset are effectively marked as true on the *in Accident* target column. For this reason, the dataset is considered to be unbalanced - too many messages marked as *false* when in comparison to the ones marked as *true*. This also explains why the models have good levels of accuracy in every subset of parameters - even if it classifies every single record as *false* (vehicles are not involved in an accident), the accuracy is very high, because only a very little subset of records are being marked inappropriately. Precisely for this fact, even when considering other metrics such as precision, recall, f-score and specificity (all had values very close to 1 (maximum value)), they do not allow, by their own, to make a proper decision on which subset of parameters performs better.
As an example, Table I shows the average results of the best performing parameters for each aggregation time when analyzing the MLPs results.
As the table shows, the results are close to perfection, but are still misleading - a more in-depth analysis of the results is necessary, with particular focus on the False Positives (FP) and False Negatives (FN). In particular, and taking into consideration the use case (detecting accidents), lowering the number of FN is of the utmost importance - it is crucial that

TABLE I
Example subset of results (MLP analysis)

Aggregation Time	Neurons	Accuracy	Precision	Reccall	F-Score	Specificity
1	32	1.000	0.999	0.998	0.998	1.000
0.5	64	1.000	0.999	0.999	0.999	1.000
0.1	32	1.000	0.998	0.999	0.998	1.000

the model is able to detect all accidents and, furthermore, detect it as soon as possible. Naturally, a high number of FP may also be a problem - it is not intended for the model to classify normal traffic situation as accidents. So, in order to conclude which parameters perform better, it is important to analyze how fast the accident is detected and the total number of false positive cases. To simplify the analysis, the results are presented regarding the best performance models for each aggregation time. All the discussed results consider the *sum* aggregation type - which performed better in all cases.
Starting with the analysis of the *1s* results, a LSTM model with two hidden layers and two dropout layers performed best - which had 64 neurons on the layers, 5 features in the input layer (*Number of vehicles, Position X, Position Y, Heading, Vehicle Width*) and 20 timesteps, as shown on Figure 2.

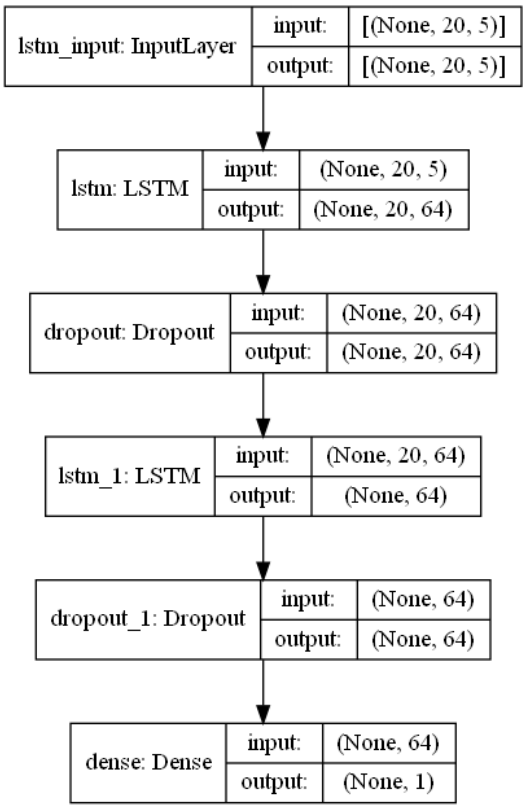

Fig. 2. LSTM Model summary

In this case, and considering a threshold of 0.9 on the output (a value greater than 0.9 equals a positive classification), 2 FP and 28 FN were found (out of 167435 entries on the test dataset). Most of the FN classifications happen right at the

beginning of the accident, which means that the model is not able to detect it immediately.

Real Value	Predicted Value
0	0.017
0	0.011
1	0.173
1	0.996
1	0.999

Fig. 3. False Negative Examples

As exemplified on Figure 3, taken from the analysis of the first accident happening in the test dataset, the model outputs an higher value on the first occurrence (compared to the when the accident is not yet happening) but still not high enough to be above the threshold of what is considered to be positive (in this case >0.9). In this case, the accident is only detected on the second instance (2s). Naturally, lowering this threshold would permit to detect the accident sooner but, unfortunately, it also results in a very high number of FP, which makes that solution unfeasible.

In summary, and considering a total of 14 accidents present on the test data, two accidents were detected in 2 seconds, ten in 3s and two in 4s - which results in an average detection of 3s. At this point, and although the model is performing in good in absolute values (very low number of FP and FN), the accident detection results are below the expectations for an automatic system.

Looking at the results when aggregating the values in a smaller time window (0.5s), the best performing model was an MLP with two dense layers (with 64 neurons each) and 8 features on the input layer (*Number of vehicles, Position X, Position Y, Speed, Heading, Acceleration, Vehicle Length, Vehicle Width*, as shown on Figure 4.

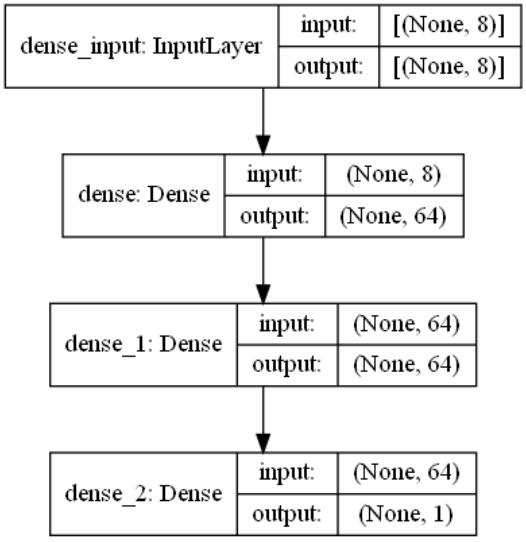

Fig. 4. MLP Model summary

Using this model, and considering a threshold of 0.95, a total of 29 FP and 12 FN were achieved. Although 29 FP may be considered good results in terms of the model's overall performance, it still may raise issues in terms of the goal of the use case: the number of FP cases is superior to the number of accidents happening on the training dataset (fourteen accidents) - which leads to the need of establishing a strategy to mitigate this issue (which was not a problem so far on the 1s case).

From the analysis of the list of FP two main conclusions were drawn. First, most of the FP (63%) happen immediately after the accident is over, and in a consecutive way. This is most likely related to the behavior of the vehicles in the mobility simulation - when the accident is over, the configuration of the vehicles on the road is still somewhat similar for some time, which may cause the model to continue to classify the entries as positive for some time in the end. To overcome this problem, it is proposed that these FP are ignored if they happen immediately after an accident has occurred. The second conclusion is that the remaining FP happen in isolated cases (there are not two consecutive FP classifications). So, to avoid FP classifications, it is established that an accident is detected only if the model classifies two consecutive record as positives. On the other hand, this also means that even if the model classifies the first entry of an accident correctly, we will only consider the accident as detected on the second entry. In other words, if the model correctly classifies the first two *0.5s* records, the accident itself will only be effectively detected in *1s*. Thus, there is a certain trade-off when implementing this strategy to overcome the problem of the FP cases - we minimize the FP problem, but we also delay the actual detection on positive cases.

The accident detections logic is illustrated on Figure 5.

t	Predicted Value	Real Value	
1s	0	0	
2s	1	0	Not considered
3s	0	0	
4s	1	0	False Positive
5s	1	0	
6s	0	0	
7s	1	1	True Positive
8s	1	1	
...			
308s	1	1	
309s	1	0	False Positive (ignored)
310s	1	0	
...			

Fig. 5. Accident Detections Logic

In this case, regarding the FN, they also tend to happen right in the beginning of the accident, similarly to the 1s case - a total of eleven accidents were detected in 1s and three in 0.5s (averaging 0.89s). So, in terms of the time needed to detect accidents, this solution performs better than the previous one, despite the model's initial worse performance in terms of FP

and FN.

Finally, the *0.1s* results are consistent with the previous cases - lowering the aggregation time results in higher FP and FN numbers (141 and 83, respectively). Similarly, FN also follow the same pattern as the other options and most of the FP happen immediately and consecutively after an accident (89%), while the remaining ones happen as isolated cases, which allows to apply the same strategy, thus mitigating the problem. This way, two accidents were detected in 0.3s, two were detected in 0.4s, one in 0.5s, three in 0.6s, two in 0.7s, two in 0.8s, one 0.9 in 1s and one 1 in 1s (averaging 0.61s). These results were obtained using exactly the same model and parameters as in the 0.5s use case.

VI. Conclusions and Future Work

This work describes the development and test of a system aimed at improving road safety, by collecting and treating ITS data through means of ML, in order to detect accidents related to VRUs (motorcycles). A simulation scenario containing accidents between passenger vehicles and motorcycles in a intersection was developed using *VEINS*, which couples mobility (*SUMO*) and communications (*OMNeT++*) simulation. The data collected from the simulation was treated and used to train a MLP model, which was also tested against different simulation data.

Results show that there is a certain trade-off between the performance of the model *per si* and the performance of the accident detection. In summary, the tested models tend to perform relatively well but they present some limitations, specially regarding the high number of FP - which may be mitigated through the use of a specific accident detection logic after the model finishes its classification. On the other hand, the use of such a detection logic of the accident also slightly delays the detection itself, so there is a certain trade-of when applying such solution. Additionally, and although the performance of the models *per si* gets worse when lowering the aggregation time (resulting in an higher absolute number of FP and FN), its capacity to detect accidents also becomes more efficient in terms of time. With the best parameter configuration, the model was able to detect every accident in 1s or less (taking 0.61s on average). This fast detection opens the possibility to trigger passive safety measures - e.g. notify surrounding vehicles of the accident; call an ambulance; notify the police; etc.

This study has some limitations which are important to take into account. First, the data used was collected by means of simulation – which naturally limits the realism and also the applicability of the solution in a real world use case. Also, the simulation scenario is somewhat simplistic (both in terms of traffic mobility and communications), so there is no indication as to its generalization capabilities to other use cases.

As future work, it is intended to study the effects of implementing passive safety measures and study its viability and capability in terms of improving the traffic flow (and also its safety) - e.g. notify surrounding vehicles of the accident and redirect them to alternative roads.

Finally, it is planned to further develop the system in order to try to predict the accidents, instead of just detecting them. Naturally, achieving such a solution would later allow for active safety measures (e.g. performing an emergency break on the passenger vehicle or notify the driver of imminent danger) which could greatly improve the *VRUs* safety on the road.

Acknowledgment

This work has been supported by national funds through FCT – Fundação para a Ciência e Tecnologia within the Project Scope: UIDB/00319/2020.

References

[1] M. Goldhammer, S. Köhler, S. Zernetsch, K. Doll, B. Sick, and K. Dietmayer, "Intentions of vulnerable road users-detection and forecasting by means of machine learning," *arXiv preprint arXiv:1803.03577*, 2018.

[2] R. Parada, A. Aguilar, J. Alonso-Zarate, and F. Vázquez-Gallego, "Machine learning-based trajectory prediction for vru collision avoidance in v2x environments," in *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 2021, pp. 1–6.

[3] M. M. R. Komol, M. M. Hasan, M. Elhenawy, S. Yasmin, M. Masoud, and A. Rakotonirainy, "Crash severity analysis of vulnerable road users using machine learning," *PLoS one*, vol. 16, no. 8, p. e0255828, 2021.

[4] M. Vilaça, E. Macedo, and M. C. Coelho, "A rare event modelling approach to assess injury severity risk of vulnerable road users," *Safety*, vol. 5, no. 2, p. 29, 2019.

[5] M. Ijaz, M. Zahid, A. Jamal *et al.*, "A comparative study of machine learning classifiers for injury severity prediction of crashes involving three-wheeled motorized rickshaw," *Accident Analysis & Prevention*, vol. 154, p. 106094, 2021.

[6] ETSI, "ETSI TR 102 638 V1.1.1 Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Definitions," ETSI, 2009.

[7] ——, "Intelligent Transport System (ITS); Vulnerable Road Users (VRU) awareness; Part 1: Use Cases definition; Release 2 ," ETSI, 2019.

[8] ——, "Intelligent Transport Systems (ITS); V2X Applications; Part 2: Intersection Collision Risk Warning (ICRW) application requirements specification," ETSI, 2018.

[9] SAE, "J2735SET - V2X Communications Message Set Dictionary Set," 2020.

[10] M.-C. Popescu, V. E. Balas, L. Perescu-Popescu, and N. Mastorakis, "Multilayer perceptron and neural networks," *WSEAS Transactions on Circuits and Systems*, vol. 8, no. 7, pp. 579–588, 2009.

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