A DL-based estimation probability approach for VRU collision avoidance

Raúl Parada[1], Rafael Corvillo[2] and Paolo Dini[1]
[1]Centre Tecnològic de Telecomunicacions de Catalunya (CTTC/CERCA)
Av. Carl Friedrich Gauss 7, 08860, Castelldefels, Barcelona, Spain
[2]Department of Computer Science, Multimedia and Telecommunications,
Universitat Oberta de Catalunya, Barcelona, 08018, Spain
{raul.parada, paolo.dini}@cttc.es, rcorvillo@uoc.edu

***Abstract*—Assisted/autonomous driving is nowadays a vivid sector for both research and industry, thanks to the advances made using artificial intelligence and the pervasive and fast communication achieved by the Fifth generation (5G) of cellular networks. A fully connected environment where traveling on public roads is done with limited (or without) human intervention may increase road safety. In this work, we present a system to detect possible collisions among vehicles and between pedestrians and vehicles with the final aim of reduce traffic accidents. Our proposal is based on a trajectory prediction algorithm plus a method to estimate the collision probability. Deep learning and Monte Carlo algorithms are used, respectively. The promising results open future research extensions.**

***Index Terms*—Collision avoidance, 5G, trajectory prediction, deep learning**

I. Introduction

5G technology has the potential to revolutionize the transportation industry and play a crucial role in the development of assisted autonomous driving. The high-speed, low-latency, and highly reliable communication capabilities of 5G networks make them ideal for supporting the massive amounts of data that autonomous vehicles need to process in real-time. With 5G, vehicles can communicate with each other, as well as with road infrastructure, to improve traffic flow, avoid accidents, and provide a safer driving experience. Additionally, 5G can provide the necessary connectivity for advanced driver assistance systems (ADAS), such as radar, cameras, and LiDAR sensors, to work together seamlessly. The implementation of 5G in assisted autonomous driving has the potential to bring about numerous benefits, including increased efficiency, improved safety, and reduced traffic congestion. Vehicular accidents are a major public health and safety issue that affect people all over the world. According to the World Health Organization, road traffic accidents are the leading cause of death for people aged 15-29 and are estimated to cause 1.35 million deaths each year. In some countries, up to 20% of road traffic deaths are pedestrians and 2-3% are cyclists. Unfortunately, vulnerable road users (VRUs) do not provide with sophisticated hardware devices such as camera and LiDAR. Thereby, it can be used uniquely geopositional data provided with most number of embedded systems. Recent works using uniquely geopositional data feeding diverse machine learning techniques [1]. This work opens a line of research on predicting trajectory predictions from common information extract by regular electronic devices. Ribeiro et al. [2] created two scenarios with SUMO. They implemented stacked unidirectional Long Short-Term Memory's (LSTMs). Although they predicted 95% of the collision, the average prediction time is above 4 seconds. According to the time required for reaction and breaking, it is about 5 seconds, hence, a prediction time above 4 seconds may not avoid a collision. Our solution aims to speed up the prediction time while providing a low precision error. Our goal is the estimation of collision avoidance probability between two road users, reducing the false alarms. Hence, our contribution includes the selection of features generated by common devices, the exploration of diverse neural network architectures to achieve a low precision error, obtaining a reasonable f1-score metric and implement a collision avoidance probability method.

II. Collision avoidance prediction system

This section presents the different phases of our work: A) data generation, B) deep learning architecture design, C) trajectory prediction and D) probability of collisions estimation.

A. Data Generation

We use the VEINS simulation framework (coupling the Simulation of Urban MObility (SUMO). We extracted and converted a neighborhood of Barcelona (Eixample) from OpenStreetMaps. It has a total of 7,914,246 records, which correspond to the trajectories of 9,981 vehicles among which 4,002 collisions occurred. Since trajectories are time-series data, generated by thousands of vehicles, we created records with trajectory windows of 20 seconds as input and the window with the following 5 seconds as output. In addition, the data set windowing procedure generates a test set with the final trajectories of the vehicles belonging to the 1000 pairs with collisions and the 1000 pairs without collisions, which have been used to evaluate the models. We decided to use the same number of collision and no collision cases for balancing. Each sample contains the following attributes:

- *time(s)*: time at a given vehicle position [1, 72000]
- *vehicle_id*: Vehicle Identification [2, 10591]
- *victim_id*: Victim identification in case of collision [-1, 10548]
- *shape_collider*: Collider vehicle type [passenger, moped, delivery, motorcycle, bus]
- *shape_victim*: Victim vehicle type [-1, passenger, delivery, moped, motorcycle, bus]
- *latitude*: Latitude geographical position [41.38698443730398, 41.39414143672337]
- *longitude*: Longitude geographical position [2.157936523084977, 2.1675456322138467]

- *speed(m/s)*: Vehicle velocity [0.0, 18.71895238439466]
- *heading(degree)*: Vehicle direction [0.0002531690841852, 359.99305154630207]
- *acceleration(m/s²)*: Vehicle acceleration [-10.0, 5.999999656410755]
- *collision*: Binary result no collision and collision [0, 1]

According to [1] and after performing further analysis, we select the attributes *latitude*, *longitude* and *heading* to train the models.

B. Deep Learning Architecture Design

With respect to our previous work [1], we have a larger number of samples, hence, we can explore deep learning techniques. For comparison, we designed and tested different deep learning architectures:

- **Simple models:** The metrics of the models with GRU and LSTM layers with different number of cells.
- **Stacked model:** Two LSTM vs GRU layers are added stacked one after the other, testing different combinations of number of units.
- **Bidirectional model:** With a bidirectional LSTM layer, the network is trained with the input sequence in both directions and the learning is concatenated.
- **Encoder-decoder model:** This type of architecture is used to problems where sequence-to-sequence predictions are made.
- **Simple model with attention mechanism:** An attention mechanism allows the neuronal network to focus attention on certain inputs and ignore others. This architecture is composed of a simple model with a layer of attention.

We configure the models with the following parameters.

- **Epochs:** which refer to the number of times the entire training set is used to train the network.
- **Batch size:** which is the size of the batches into which the training set is divided.
- **Optimizer:** The optimizer used is the Adam algorithm, which is an adaptive gradient descent method, and a learning rate of 0.001 is used.
- **Early stopping:** this mechanism is used to stop training when the model stops learning.
- **ReduceLROnPlateau:** mechanism is used to reduce the learning speed when the model stops improving. The specific values chosen for these parameters were determined through initial testing and adjusted to find a balance between performance and training time.

After completing the training of each model, a series of metrics are collected to analyze and compare the models:

- **Mean Squared Error (MSE)**: Calculates the sum of the square of the errors of the predicted values with respect to the actual ones. It is used as a function of losses in training, always seeking to minimize its value. This metric is the most commonly used in regression problems.
- **Mean Absolute Error (MAE)**: Calculates the difference between the predicted and actual values, and has been used as a metric during training.
- **Execution time (s)**: Indicates the total time, in seconds, that the training has lasted.
- **Emissions CO_2 (kg)**: Indicates the CO_2 emissions, in kilograms, that have occurred during the execution of the training code. We use the python library CodeCarbon to perform this measurement [3].

C. Trajectory Prediction

For the detection of a collision, a method will be used that calculates the intersection of the polygons of each vehicle. This method consists of the following steps:

1) Predict the trajectory of the next 5 seconds of both vehicles.
2) Extract the predicted latitude, longitude, and direction for the fifth second of both vehicles, since the collision occurs in the last second of the sequence.
3) From the three previous attributes and the dimensions of the vehicle, the points of the rectangle that contains each vehicle are calculated.
4) Generate the polygon of each vehicle from the points calculated in the previous step.
5) Check if the intersection of the polygons is greater than zero, that is, $\upsilon_1 \cap \upsilon_2 \neq \varnothing$, where υ_1 is the polygon of vehicle 1, υ_2 is the polygon of vehicle 2 and $\varnothing$ is the set empty.

We executed the above steps to all 1000 pairs of vehicles that collide and the 1000 pairs of vehicles that do not collide. From those runs will get a confusion matrix with a total of true positives (TP), an occurred collision was detected correctly; false negatives (FN), an occurred collision was not detected; false positives (FP), a non occurred collision was detected; and true negatives (TN), a non occurred collision was not detected. We use the f1-score to evaluate our models in which the above metrics are involved.

D. Probability Estimation of Collisions

To calculate this probability of collision, we developed the algorithm 1 inspired from [4]. This algorithm is a Monte Carlo algorithm which estimates the probability of collision from the probabilistic models of the vehicles. The algorithm represent the probability density function (PDF) of both vehicles' predicted trajectories. We compute the PDF of the 5-second trajectories of each vehicle, and then N random samples are drawn from these PDFs and check if a collision is detected (c.f. II-C). Finally, the collision probability is calculated by dividing the total number of detected collisions by the total number of samples.

Algorithm 1 Probability estimation of collision between two vehicles

Require: 5 seconds trajectory from each vehicle $x_\upsilon = (lat_t, long_t, \theta_t)$.
Ensure: Collision probability P_{coll}.

```
P_coll(υ1, υ2) ← 0
for j ← 1 to N do
    x_υ1 ← randc(x̂_υ1, Σ_υ1)
    x_υ2 ← randc(x̂_υ2, Σ_υ2)
    if υ1 ∩ υ2 ≠ ∅ then
        P_coll(υ1, υ2) ← P_coll(υ1, υ2) + 1
    end if
end for
P_coll(υ1, υ2) ← P_coll(υ1, υ2) / N
return P_coll(υ1, υ2)
```

Model	Training time(s)	Emissions CO_2 (kg)	Median error (second 5) (m)	F1-score
Simple GRU (64 cells)	6.317	0.044183	12.088	0.064
Simple LSTM (64 cells)	8.314	0.058144	12.463	0.087
Stacked GRU (32-32 cells)	6.212	0.017822	12.216	0.083
Stacked LSTM (32-32 cells)	14.785	0.055411	12.224	0.055
Bidirectional (32 cells)	4.785	0.008716	12.504	0.073
Encoder-decoder (32 cells)	5.412	0.032157	12.188	0.055
LSTM model (32 cells) with attention mechanism (64 cells)	5.076	0.009246	11.947	0.075

TABLE I: The median error is similar along all models. However, bidirectional and attention mechanism provides the best metrics of emissions and F1-score

III. Results and conclusions

This section collects all the executions using the different neural network architectures. Unfortunately, due to space limitation we cannot add all the results, only the last results will be provided [5].

Table I summarizes the results of: training time in seconds, CO2 emission in kilograms, median error between the real position and the prediction in meters in the fifth second to foresee and the f1-score, from all the experimentation runs. We can observe how the f1-score values are similar in all cases, however, both the bidirectional and using the attention mechanism model reach almost the 10%. In addition, those models result in a similar CO2 emission and median error. As mentioned in [2], the SUMO environment does not simulate collisions by default. Hence, some expected collisions might not be real. As a consequence, the number of simulated collisions end up being much lower than programmed. Assuming this fact, we decide to apply the algorithm to estimate the probability of collision. Mention that N in algorithm is user-configurable. The intuition within the variable N is that the larger it is, more precise would be the probability with the contrast that the algorithm will take longer time. Since our system must predict the collisions with a reasonable time to be useful. For instance, we cannot predict a position 5 seconds in the future when the computation time takes already 4 seconds. This fact will only allow a difference of one second to react and brake in case of a possible real vehicular situation. Thereby, N with a value of 300 balances an optimal probability collision estimation and computation time. In this case, the computation time is about 930 ms, resulting on 3 ms per simulation.

Figure 1 shows the collision probabilities with N equal to 300. We can observe how an average probability of 1.997% is obtained. We set a threshold to decide when a warning message should send to the drivers to reduce false alarms but protecting the drivers. The 25th percentile of the pairs of vehicles that collide has been chosen as the threshold, that is, a probability of 1.6%. From the results shown in the Figure 1 and the threshold, we verify that alarms will only be generated for 15.2% of the vehicles that collide and for 8.7% of vehicles that do not collide. We use As mentioned in [2] the SUMO environment does not provide a collision features by default, therefore, configured collisions might not occur resulting on a lower number of expected collisions. This fact might have influenced the f1-score values (c.f. Table I).

This work presented an estimation probability collision avoidance system to reduce VRU accidents on 5G-equipped road environments. We performed a large comparison of deep learning techniques with median error in the fifth second about 12 meters predicted in 3 ms beside of above 4 seconds [2]. Although the f1-score was not the ideal, we get a good result with the amount of data where vehicles remain static. For this reason, we plan but no limit to increase the dataset to balance the number of speed zero samples, optimize the programming environment to enlarge the value of N to improve the estimation probability and explore advanced neural network architectures.

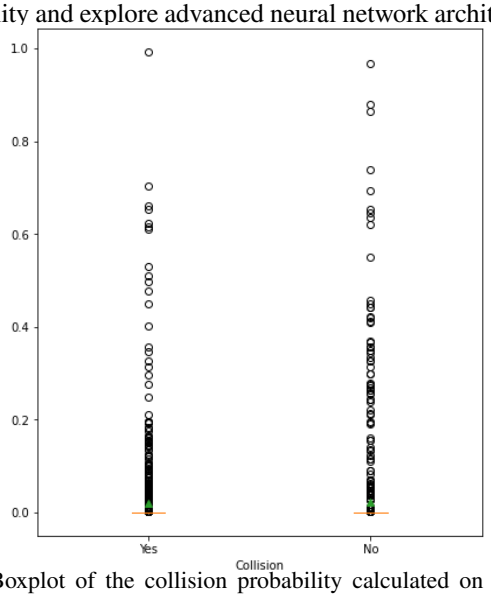

Fig. 1: Boxplot of the collision probability calculated on predicted trajectories

Acknowledgment

We would like to thanks to Eudald Llagostera and Adrià Pons from Fundació i2CAT for the generation of the dataset from SUMO. This work has been partially funded by the 5GMED project (H2020-No951947).

References

[1] 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)*, 2021, pp. 1–6.

[2] B. Ribeiro, M. J. Nicolau, and A. Santos, "Using machine learning on v2x communications data for vru collision prediction," *Sensors*, vol. 23, no. 3, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/3/1260

[3] V. Schmidt, K. Goyal, A. Joshi, B. Feld, L. Conell, N. Laskaris, D. Blank, J. Wilson, S. Friedler, and S. Luccioni, "CodeCarbon: Estimate and Track Carbon Emissions from Machine Learning Computing," 2021.

[4] A. Lambert, D. Gruyer, and G. Saint Pierre, "A fast monte carlo algorithm for collision probability estimation," in *2008 10th International Conference on Control, Automation, Robotics and Vision*, 2008, pp. 406–411.

[5] R. Corvillo, "vehicles_collision_detector," https://github.com/rcorvial/vehicles_collision_detector, 2023.