Driver Behaviour Analysis based on Deep Learning Algorithms

Ion Cojocaru
University of Craiova,
Craiova, Romania
13 A.I.Cuza Street, 200585
Craiova, Romania
kojocaru.ivan@gmail.com

Paul-Stefan Popescu
University of Craiova,
Craiova, Romania
13 A.I.Cuza Street, 200585
Craiova, Romania
stefan.popescu@edu.ucv.ro

Cristian Mihaescu
University of Craiova,
Craiova, Romania
13 A.I.Cuza Street, 200585
Craiova, Romania
cristian.mihaescu@edu.ucv.ro

ABSTRACT
This paper presents a system designed and implemented for driver behaviour analysis using deep learning methods. The system is described at a high-level view along with the dataset collected and used for experimental results. The system is straightforward to use and depends only on the smartphone for both collecting data and estimating driver behaviour. The experiments were conducted using three deep learning approaches, each of them based on LSTM networks and showing average results on the standard dataset and very good results when lowering the number of classes from the dataset. The novelty of this paper comes from the fact that it presents a new dataset which can be collected and improved using any smartphone and the approach of using these algorithms for this specific context.

Author Keywords
Driving behaviour; machine learning, dataset

ACM Classification Keywords
H.5.m. Information interfaces and presentation: Miscellaneous.

General Terms
D.2.2: Design tools and techniques. H5.2 User interfaces.

DOI: 10.37789/rochi.2022.1.1.18

INTRODUCTION

Traffic nowadays tends to increase in every corner of the world, and with-it different driving approaches appear. The motivation for automatic estimation of driving behaviour comes from different directions as it may serve to find aggressive drivers or, in our context, offer better insight into the available drivers. Driving behaviour classification can bring benefits more than preventing finding an aggressive driver, but also it may satisfy the user's needs as there may be clients who would want, for several reasons, to take an aggressive/fast driver instead of a slow one. Another motivation for such a system is that ride-sharing applications and systems became more popular and used often but this module can be used also for taxi drivers or any driver who have an android device.

This paper presents a driving behaviour estimator module designed and build for a custom driver's management system. The aim of the driving management system is a ride sharing application which is composed by two android applications and one server-side application each of them with a clear scope. The android applications are one for the client and one for the driver allowing the client to ask for a ride from his/her place to a specific destination and the driver allows to navigate and to bring, compute the cost of the ride along with other regular functionalities and two more specific functionalities: the first one allows for data collection and the second one allows driving behaviour estimation. The driving behaviour estimation is done offline, on the driver application thanks to the *tflite* model which is integrated into the application so there is no need to have an active internet connection when the classification is performed. Based on the deep learning models used, it is enough to drive more than 10 seconds to have a driving behaviour estimation and as the driver drives longer the estimation becomes more robust. The second feature integrated into the driver which is relevant for this paper is the data collection module which creates a labelled *.csv (Comma separated value)* file. This file can be used for training or model improvement so we can continuously improve the models as we collect more and more data.

The driver behaviour estimation module is based on deep learning techniques and was built using TensorFlow. The motivation for choosing TensorFlow as framework for this module comes from its flexibility on building models and also the possibility of exporting the model for using in other applications, like *tflite* for android which decouples the python module used for training and model optimization from the driver's application which was built in java.

The system was built with the aim to have a high usability and to work in many driving conditions as long as a android device is inside the car and the application is running. In order to build the dataset used for this system we also used an android device and found some things that needs to be considered when collecting data or estimating the driving behaviour. The first thing is that the device needs to be in a fixed place, putting it on the car's board, chair or places that allows it to move may alter the results because of part of the force applied to the device would be absorbed by the

movement. The same rule applies to the driver's device used for driving behaviour estimation as it needs to be in a fixed position in any place in the car including windshield mounts.

In order to build a system which would adapt in many situations the application will use the data collected only from gyroscope and accelerometer built into the device. Another sensor that could have been implemented would be the speed from the GPS module but that may not be that relevant as the maximum speed is constant in cities and there may be calm drivers which would drive at maximum legal speed and aggressive drivers which would also comply to the driving rules but the difference would come from the way they accelerate or hit the brake pedal or how they manage to turn the vehicle or change the lanes. Aggressive driving can be performed at different speeds and does not necessarily depend on how fast you drive. Another motivation for not including the speed into the dataset is that we need to make a generic system which will provide relevant results in different cities or countries and the maximum speed allowed may differ depending on the rules and laws specific from that country.

RELATED WORK

There are many research directions which were explored previous to building the system and writing this paper. The first direction was regarding the datasets used for predicting the driving behaviour and in this area, there are plenty off approaches and papers but most of them have a different approach or drawbacks.

One interesting paper [1] uses machine learning for driving risk prediction and they used a variety of attributes for driving behaviour analysis. Through in-depth analysis of driving behaviour data, the authors use machine learning methods to analyze and predict driving risk, more exactly SVM [2], neural networks and random forest [3]. Their experiments shows that it is necessary to take detailed analysis into consideration and their dataset is not public so only the knowledge and methods presented in the paper are valuable lacking reproducibility or the possibility to integrate in a bigger system.

Another paper [4] which presents an approach for predicting the driving behaviour but uses a more social approach combined with machine learning techniques shows how personality, age, and power of car can influence an aggressive driving behaviour. In the study, a total of 154 male motorists completed two subscales of the Hypermasculinity Inventory as a measure of macho personality and self-reports of aggressive driving behaviour based on the Driver Behaviour Questionnaire [5]. They also provided information about their age, annual mileage, horsepower of their car, and features that had guided their choice of a car. A multiple regression analysis showed that each of the predictors was significantly related to aggressive driving. Their approach was relevant for finding aggressive driver based on their data and differs than ours which finds their driving style while driving. For a ride-sharing application their approach may work or not, depending on the context and we should also take into consideration that a driving style may change over time.

A driver performance model based on machine learning was published in 2018 [6] and the authors present an approach for building a novel driver performance model, which is unique for every driver. Their driver is modelled using machine learning algorithms, namely artificial neural network and adaptive neuro-fuzzy inference system. Each model is trained and validated with the data collected during the real-time driver-in-the-loop experiment on a vehicle simulator for each driver separately. This approach of using vehicle simulators bring consistency in results but may have drawbacks compared to real world driving. In their study, 18 participants contributed to the experiment. Although the prediction accuracy of the models depends on the algorithm specifications, the artificial neural network was slightly more accurate in driver performance prediction comparing to the adaptive neuro-fuzzy inference system.

Another newer paper [7] explores spatio-temporal attributes and their impact on predicting taxi drivers' risky and aggressive driving behaviour. Their study aimed to comprehensively investigate different traffic violations using spatial analysis and machine learning methods in the city of Luzhou, China. Results revealed that over-speeding was the most prevalent violation type observed in the study area. Frequency-based nearest neighbourhood cluster methods in Arc map Geographic Information System (GIS) were used to develop hotspot maps for different violation types that are vital for prioritizing and conducting treatment alternatives efficiently. Finally, different machine learning (ML) methods, including decision tree, AdaBoost with a base estimator decision tree, and stack model, were employed to predict and classify each violation type. The proposed methods were compared based on different evaluation metrics like accuracy, F-1 measure, specificity, and log loss. Prediction results demonstrated the adequacy and robustness of proposed machine learning (ML) methods.

An interesting method for creating a dataset is presented in paper [8] which addresses the problem of Collecting and Processing a Self-Driving Dataset. In their paper, they provide a guideline going through all the steps from configuring the hardware setup to obtaining a clean dataset. They describe the data collection scenario design, the hardware and software employed in the process, the challenges that must be considered, data filtering and validation stage. In their belief, having a clean and efficient process of collecting a small but meaningful dataset has the potential to improve benchmarking autonomous driving solutions, capturing local environment particularities.

A paper that describes a dataset is also [9] which classifies driving behaviours into five dimensions, which are speeding, improper overtaking, mobile phone use while

driving, tail-gating and disobeying traffic lights. They perform a quantitative study with a sample size of 160 drivers consisting of residents in a suburban of Selangor, Malaysia. A stratified random sampling method was adopted to identify the respondents. Data analysis was presented in the form of descriptive statistics and tables. The findings show that most respondents agreed that they have driving behaviours that involve improper overtaking, tail-gating and disobeying traffic lights.

SYSTEM DESIGN

In order to understand the driving behaviour estimation, we need to take a closer look at the whole system, which is composed of different components that leads to the whole ride-sharing system.

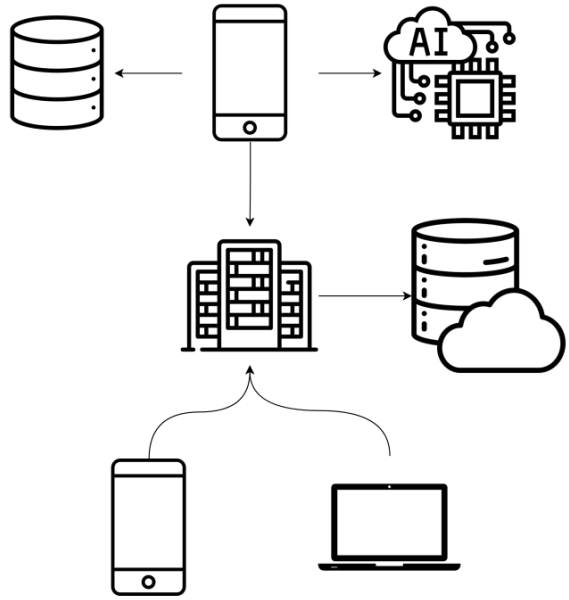


Figure 1. High-level system overview

Figure 1 presents the components of the system, which integrates the driving behaviour component. The whole system is composed of a web application, the server module, two mobile applications (one for the client and one for the driver, two databases, one in the cloud and one in for the mobile application and the driving behaviour module, which is used for model training and prediction.

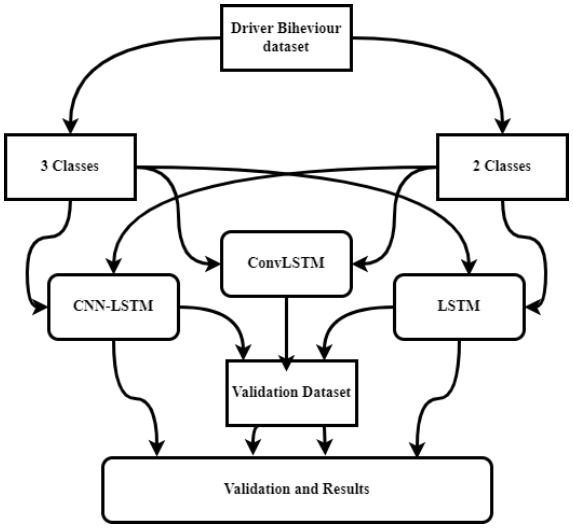


Figure 2. The driving behaviour estimation system

Figure 2 presents the overview for the driving behaviour dataset, and based on this image, everything starts from the dataset, which was initially collected having three classes: slow, normal and aggressive and then based on the experiments performed, we merged slow and normal and obtained a new dataset with two classes which are offering better accuracy. In order to explore which is the best model, we trained three different models: LSTM [10], ConvLSTM [11] and CNN-LSTM [12] and evaluated them using a different dataset which was collected on a different route. The results obtained using this approach can be seen in the experimental results section.

The motivation for choosing LSTM and other LSTM derivations comes from the fact that one single event may not be relevant for a driving style; more exactly, it can be a strong brake or maybe the driver hits an asphalt pothole which leads to considerable accelerometers value but then everything goes to normal. Using an LSTM implies a succession of steps taken into consideration and a timeframe which is used for classification so the results offered by such a system may be more robust and reliable. Of course, there are plenty of machine learning or deep learning methods that can be used to predict driving behaviour, but we need to take into consideration that when driving, there are many events that may happen and would make the driver change his driving style for s short period of time like passing an obstacle which appears on the road or drive on a clogged road where even the most aggressive driver can not have a driving style different than slow.

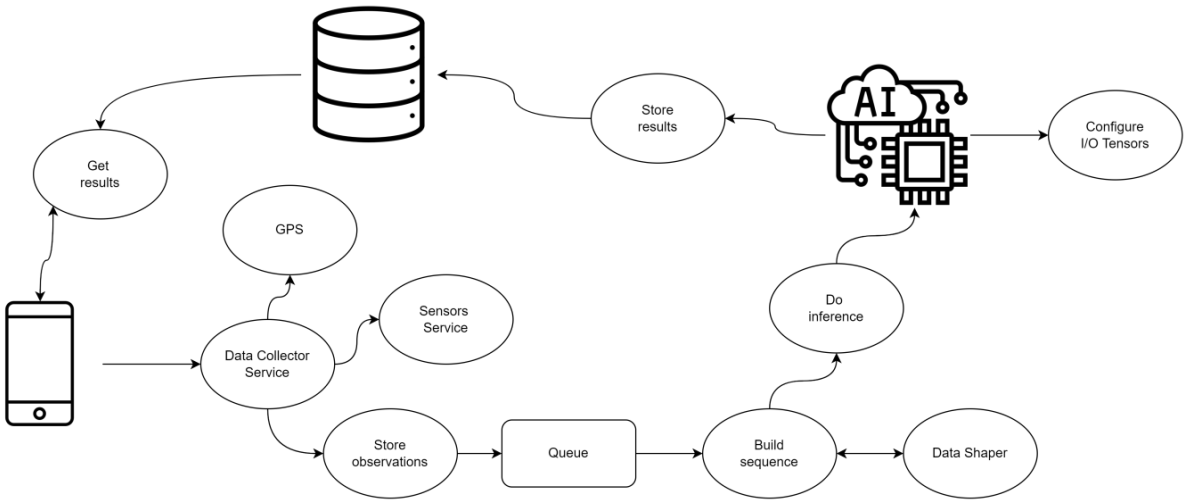


Figure 3. Driver's mobile application architecture

Figure 3 presents the driver's mobile application architecture, which explains how the driving behaviour system works; this figure goes to an overview of the whole application, not only the classification system. On the mobile phone is a data collection service which uses the sensors and stores the observations; based on the stored observations, a sequence can be made for further classification.

The dataset used in the application consists of 8 features:

- 3 for acceleration on each axis (X, Y and Z in meters per second squared)
- 3 for rotation (X, Y, Z axis in degrees per second (°/s))
- Classification label (SLOW, NORMAL, AGGRESSIVE)
- Timestamp

The data was collected in samples (2 samples per second), the gravitational acceleration was removed, and only two main sensors were used: Accelerometer and Gyroscope.

RESULTS

Before presenting the actual results of the driving behaviour system, we need to show the dataset used for training and then for validating the results. The dataset is public and available on Kaggle[1] for downloading and contribution with notebooks. The whole system is available on GitHub, along with the application used for creating the dataset[2].

```
AccX,AccY,AccZ,GyroX,GyroY,GyroZ,Class,Timestamp
0.0,0.0,0.0,0.059406646,-0.17470746,0.101938136,NORMAL,3581629
-1.6248639,-1.0824918,-0.20418262,-0.02855795,0.05131268,0.13553573,NORMAL,3581630
-0.5946599,-0.122409865,0.2205019,-0.019394971,-0.029321533,0.08788824,NORMAL,3581630
0.7384782,-0.22845554,0.66773224,0.069791354,-0.029932396,0.054901514,NORMAL,3581631
0.101741076,0.7775676,-0.066729546,0.030695979,-0.0036651916,0.054901514,NORMAL,3581631
0.15846956,0.34589112,0.3552742,0.021533,0.115453534,0.014584408,NORMAL,3581632
0.078171015,-0.34917977,0.27065182,0.03436117,-0.030543262,0.05062546,NORMAL,3581632
-0.66274977,-0.46277535,-0.09979725,0.00870483,-0.009773844,0.046960264,NORMAL,3581633
1.5441127,-0.13220355,0.98183155,-0.011453724,-0.040317107,-0.015958855,AGGRESSIVE,3582522
-0.62998724,-0.8365323,-1.6980171,0.09666943,-0.108123146,-0.062384613,AGGRESSIVE,3582522
-0.29793346,-1.1543016,-2.4697137,0.11133019,-0.20525073,0.48006374,AGGRESSIVE,3582523
0.4789641,-2.1047263,1.5465002,-0.060322944,0.27855456,0.11293371,AGGRESSIVE,3582523
-1.3421266,-1.009614,0.17931366,0.018478673,-0.029321533,-0.05505423,AGGRESSIVE,3582524
0.20869446,0.2011531,1.3307877,-0.110413894,-0.1160644,0.026801713,SLOW,3583316
0.93586683,-0.48866272,-0.68681145,-0.017562376,0.030543262,0.017027868,SLOW,3583317
-0.38256752,0.05143237,-0.26056576,-0.030390546,0.023823744,0.035353824,SLOW,3583317
0.07829058,-0.39971197,-1.2749577,0.016035212,-0.056199603,-0.04466952,SLOW,3583318
1.009754,0.6672493,-1.6077585,0.11133019,0.09468412,0.030466905,SLOW,3583318
0.22656584,-2.1912546,-0.05291748,0.03924809,-0.35185838,-0.18761198,SLOW,3583319
```

Figure 4. Dataset sample

Figure 4 represents a sample of 19 instances from the dataset, along with the features names. There are two datasets available on Kaggle: one for training and one for testing. However, despite the fact that they are different and collected on different routes, each can be used for training, or we can even combine them into a larger one. Still, in this case, the results may be biased.

The results are divided into two sets: one using three classes and one using two types (slow and regular merged). The timestamp is used for logging purposes and helps us with the training process because we need to define timeframes when training the algorithms, as most are based on LSTM networks.

Results for three classes

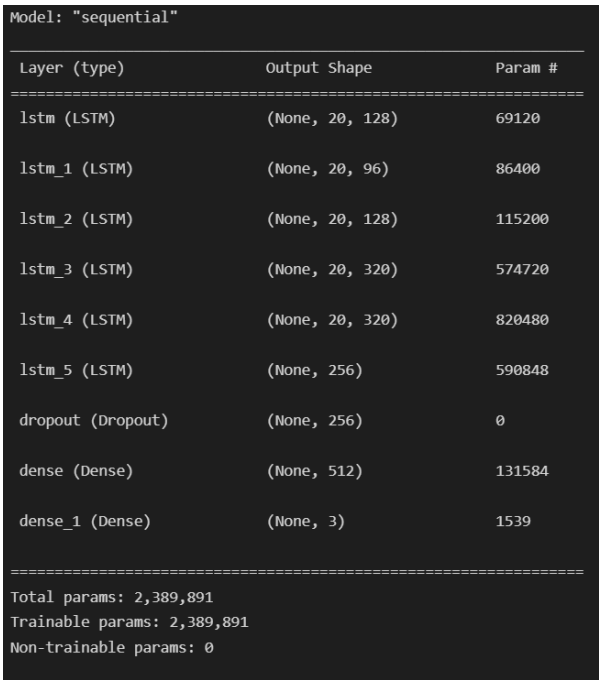
Each set of results is divided into three approaches which use different deep learning algorithms.

[1] Driving behaviour dataset:

https://www.kaggle.com/datasets/outofskills/driving-behavior

[2] Driver's app:

https://github.com/OutofSkills/AndroidDriverApp



```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm (LSTM)                 (None, 20, 128)           69120

 lstm_1 (LSTM)               (None, 20, 96)            86400

 lstm_2 (LSTM)               (None, 20, 128)           115200

 lstm_3 (LSTM)               (None, 20, 320)           574720

 lstm_4 (LSTM)               (None, 20, 320)           820480

 lstm_5 (LSTM)               (None, 256)               590848

 dropout (Dropout)           (None, 256)               0

 dense (Dense)               (None, 512)               131584

 dense_1 (Dense)             (None, 3)                 1539

=================================================================
Total params: 2,389,891
Trainable params: 2,389,891
Non-trainable params: 0
```

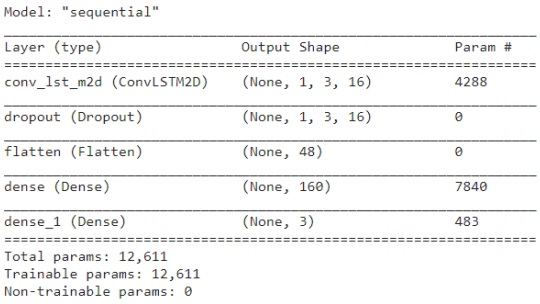
Figure 5. LSTM network configuration for 3 classes

Figure 5 presents the LSTM network structure used to achieve the results presented in the table below (table 1).

Window size	Timeframe covered (½ * window size)	Accuracy
16 instances	8 sec	40.32%
12 instances	6 sec	54.88%
8 instances	4 sec	42.28%
6 instances	3 sec	40.85%

Table 1. Results for LSTM with 3 classes

The first table (Table 1) presents the results obtained using a standard LSTM network with a variation of 16,12, 8 and 2 instances, along with the timeframe covered and the obtained accuracy. In this case, the optimum timeframe for computing the driving style is 6 sec, which offers 54.88% accuracy.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv_lst_m2d (ConvLSTM2D)    (None, 1, 3, 16)          4288
_________________________________________________________________
dropout (Dropout)            (None, 1, 3, 16)          0
_________________________________________________________________
flatten (Flatten)            (None, 48)                0
_________________________________________________________________
dense (Dense)                (None, 160)               7840
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 483
=================================================================
Total params: 12,611
Trainable params: 12,611
Non-trainable params: 0
```

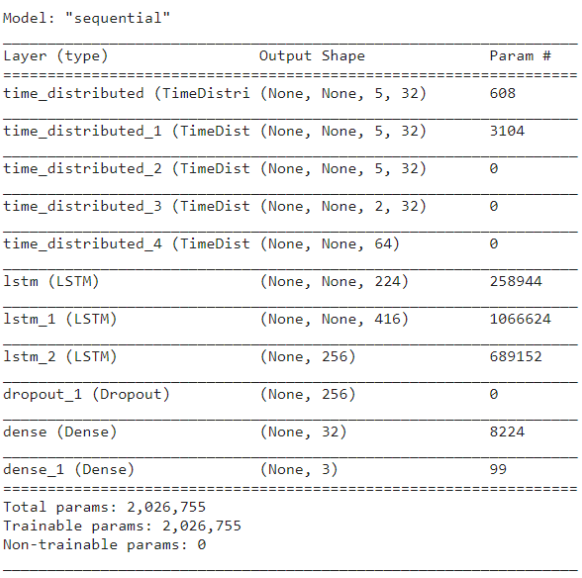
Figure 6. ConvLSTM structure

Figure 6 presents the structure of the ConvLSTM network used for obtaining the results presented in table 2.

Window size	Timeframe covered (½ * window size)	Accuracy
18 instances	9 sec	59.26%
16 instances	8 sec	58.06%
8 instances	4 sec	51.16%

Table 2. Results for ConvLSTM with 3 classes

The next set of results is presented in Table 2, which shows the results for 9,8 and four seconds. We can observe a performance increase over the previous LSTM approach; in this case, the best accuracy was correlated with the biggest timeframe. We couldn't evaluate for timeframes longer because of the dataset dimension.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
time_distributed (TimeDistri (None, None, 5, 32)       608
_________________________________________________________________
time_distributed_1 (TimeDist (None, None, 5, 32)       3104
_________________________________________________________________
time_distributed_2 (TimeDist (None, None, 5, 32)       0
_________________________________________________________________
time_distributed_3 (TimeDist (None, None, 2, 32)       0
_________________________________________________________________
time_distributed_4 (TimeDist (None, None, 64)          0
_________________________________________________________________
lstm (LSTM)                  (None, None, 224)         258944
_________________________________________________________________
lstm_1 (LSTM)                (None, None, 416)         1066624
_________________________________________________________________
lstm_2 (LSTM)                (None, 256)               689152
_________________________________________________________________
dropout_1 (Dropout)          (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 32)                8224
_________________________________________________________________
dense_1 (Dense)              (None, 3)                 99
=================================================================
Total params: 2,026,755
Trainable params: 2,026,755
Non-trainable params: 0
_________________________________________________________________
```

Figure 7. CNN-LSTM network structure

The last explored model is presented in Figure 7, and its results are presented in the table below. This approach is

the most reliable model being quite independent of the window size.

Window size	Timeframe covered (½ * window size)	Accuracy
16 instances	8 sec	48.39%
12 instances	6 sec	48.78%
8 instances	4 sec	50.0%
6 instances	3 sec	51.22%

Table 3. Results for CNN-LSTM with 3 classes

The last set of results (Table 3) obtained for the three classes approach is the CNN-LSTM network, which offers consistent results for all the timeframes covered. Even though CNN-LSTM is a very consistent approach, we can see that it is best to be used with small timeframes.

Results obtained for two classes

In order to obtain better results and obtain a better prediction, we explored the possibility of merging class slow with class normal and trying to find the aggressive drivers. To do this, we run the same configuration of classifiers but on the modified dataset.

Window size	Timeframe covered (½ * window size)	Accuracy
16 instances	8 sec	80.37%
12 instances	6 sec	82.15%
8 instances	4 sec	75.91%
6 instances	3 sec	63.80%

Table 4. Results for LSTM with 2 classes

Table 4 presents the results obtained using the same LSTM network used in Table 1 but with those two classes merged. The best accuracy is obtained still using 12 instances, but for each window size, we get increased accuracy.

Window size	Timeframe covered (½ * window size)	Accuracy
18 instances	9 sec	81.48%
16 instances	8 sec	85.37%
8 instances	4 sec	83.87%
6 instances	3 sec	67.86%

Table 5. Results for ConvLSTM with 2 classes

Table 5 presents the results obtained using ConvLSTM but considering 2 classes; we can observe an increase in accuracy over the results presented in Table 2. In this case, even going as low as 6 instances, we get better accuracy than using 3 classes.

Window size	Timeframe covered (½ * window size)	Accuracy
16 instances	8 sec	87.10%
12 instances	6 sec	91.94%
8 instances	4 sec	83.70%
6 instances	3 sec	74.86%

Table 6. Results for CNN-LSTM with 2 classes

Table 6 presents the results obtained for CNN-LSTM but with merged datasets. Table 6 is the correspondent for Table 3, and the accuracy differences are significant, offering superior accuracy and the best performance for predicting driving behaviour. The best window size, in this case, is composed of 12 instances which correspond to 6 seconds of logged data.

CONCLUSION

This paper presented a driving behaviour estimation using deep learning techniques and a custom-built dataset. For better usability and usefulness, the system was integrated into a ride-sharing application so the customers would know in advance more details about the driver.

The experiments revealed that a better accuracy could be obtained when using two classes rather than three classes, and using CNN-LSTM we can get the best results for two classes while ConvLSTM offers the best results for three classes. In any case, the two variations of LSTM (Conv and CNN) explored in the experiments outperformed the classical LSTM network.

REFERENCES

1. Wang, Y., Xu, W., Zhang, Y., Qin, Y., Zhang, W., & Wu, X. (2017, November). Machine learning methods for driving risk prediction. In Proceedings of the 3rd ACM SIGSPATIAL workshop on emergency management using (pp. 1-6).
2. Noble, W. S. (2006). What is a support vector machine?. Nature biotechnology, 24(12), 1565-1567.
3. Schonlau, M., & Zou, R. Y. (2020). The random forest algorithm for statistical learning. The Stata Journal, 20(1), 3-29.
4. Krahé, B., & Fenske, I. (2002). Predicting aggressive driving behaviour: The role of macho personality, age, and power of car. Aggressive Behaviour: Official

Journal of the International Society for Research on Aggression, 28(1), 21-29.

5. Lawton, R., Parker, D., Manstead, A. S., & Stradling, S. G. (1997). The role of affect in predicting social behaviours: The case of road traffic violations. Journal of applied social psychology, 27(14), 1258-1276..
6. Aksjonov, A., Nedoma, P., Vodovozov, V., Petlenkov, E., & Herrmann, M. (2018). A novel driver performance model based on machine learning. IFAC-PapersOnLine, 51(9), 267-272.
7. Zahid, M., Chen, Y., Khan, S., Jamal, A., Ijaz, M., & Ahmed, T. (2020). Predicting risky and aggressive driving behaviour among taxi drivers: do spatio-temporal attributes matter?. International journal of environmental research and public health, 17(11), 3937.
8. Nica, A. C., Trsăcău, M., Rotaru, A. A., Andreescu, C., Sorici, A., Florea, A. M., & Bacue, V. (2019, May). Collecting and processing a self-driving dataset in the UPB campus. In 2019 22nd International Conference on Control Systems and Computer Science (CSCS) (pp. 202-209). IEEE.
9. You, H. W., Rahman, A. A., & Dwisatrya, L. H. H. (2020). Dataset of driving behaviours in Selangor, Malaysia. Data in brief, 31, 105783.
10. Han, Q., Hu, X., He, S., Zeng, L., Ye, L., & Yuan, X. (2018, November). Evaluate good bus driving behaviour with lstm. In International Conference on Internet of Vehicles (pp. 122-132). Springer, Cham.
11. Chen, X., Xie, X., & Teng, D. (2020, June). Short-term traffic flow prediction based on convlstm model. In 2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC) (pp. 846-850). IEEE.
12. Patel, E., & Kushwaha, D. S. (2022). A hybrid CNN-LSTM model for predicting server load in cloud computing. The Journal of Supercomputing, 78(8), 1-30.