An LSTM-Based Approach for Driving Behaviour Classification

Alberto Pingo
Department of Computer Engineering
Polytechnic Institute of Leiria
Leiria, Portugal
2202145@my.ipleiria.pt

João Castro
Department of Computer Engineering
Polytechnic Institute of Leiria
Leiria, Portugal
2201781@my.ipleiria.pt

Mentors
Anabela Moreira Bernardino
Paulo Jorge Gonçalves Loureiro
Sílvio Priem Mendes

The aim of this work is to explore, analyze and classify driving behaviours using recurrent neural networks. By collecting data from a mobile application, this study seeks to identify driving patterns and train a Long Short-Term Memory model to provide a precising and reliable classification for driving patterns.

The process starts with data collection and preprocessing. **After** *that, during the development of an LSTM-based neural network, the train is fed. The model architecture merges Convolutional Neural Networks with LSTM to enhance the detection of temporal and spatial dependencies existing in driving data. The hybrid model is trained and validated using an 80/20 split of the dataset to ensure its robust performance evaluation.*

Experimental results demonstrate that the proposed methodology has potential for further improvement of management of traffic, reducing accidents, and enhancing transportation safety. This paper is concluded with a discussion about the results regarding high accuracy and reliability of the model in classifying driving behaviors, like the possibility of integration of such systems into Intelligent Transportation Systems in the future.

Keywords: Artificial Intelligence, Neural Networks, LSTM, RNN, Driving Classification

I. INTRODUCTION

Artificial intelligence has caused several industries to change or evolve for the better, and the automotive sector is no exception where the classification of different driving behaviours has become prominent in developing road safety.

Thise work aims to harness the power of neural networks to analyse and classify driving behaviour. Knowledge of driving behaviour is cardinal in improving road safety and developing more Advance Driver-Assistance System and autonomous vehicles. This work recognizes and classifies the driving behaviour concerning acceleration, breaking, and driving style by using data off a mobile application.

The importance of driving behaviour classification is growing not only within the automotive sector but also within such sectors as transport and logistics, where insight into this behaviour would mean better fleet flow, reduced fuel consumption, and general operational efficiency. Detailed driving behaviour analysis can also be very useful in aspects related to urban planning and public safety by providing information for better traffic management and road design based on real driving patterns.

Our implementation focuses on not just training a model but also processing the data to correctly classify driving behaviours. In our approach, a smartphone has been used because most of today's phones come with state-of-the-art sensor hardware: just an accelerometer, gyroscope, and GPS. We have done rigorous processing on the collected data to get the best possible results for our model. This includes segregating data with respect to the types of manoeuvres, after which normalization and labeling of the dataset are done. All these preprocessing techniques enable one to get the most informative features while reducing noise, which would impair high accuracy for the classification model. LSTMs are applicable in this task since they help in capturing intrinsic dependencies and correlations in time-series data acquired during real driving sessions. By this, we would be able to classify different driving behaviours, either normal driving or an aggressive one.

Otherwise, our key aim is to get hold of the best LSTM that is to be vigorously trained so as to ensure a very accurate classification of driving patterns.

II. RELATED WORK

Several architectures have been proposed and evaluated for their effectiveness in driving behaviour classification. Previous studies have recognized that LSTMs perform very well in various sequence prediction tasks. In driving behaviour, such models have been able to achieve high accuracy, precision, and recall classifying various driving behaviours. For instance, Saleh et al. suggested an approach using stacked LSTM for the classification of driving behaviour according to sensor data fusion, and tested it with the UAH-DriveSet dataset. Their model efficiently classified the behaviours under three headings: normal, aggressive, and drowsy driving. Moreover, combining LSTMs with other neural network models has turned out to improve the classification performance. Deo and Trivedi presented an

LSTM model for interaction-aware motion prediction of surrounding vehicles on freeways. Their model showed significant reduction in prediction error with interaction using the NGSIM US-101 and I-80 datasets, thus being effective in vehicle trajectory prediction. Khodairy and Abosamra proposed a deep learning-based solution for driving behavior classification using the optimized Stacked-LSTM model with signals from the smartphone-embedded sensors. The authors developed models for three-class classification distinguishing between normal, drowsy, and aggressive driving behaviors and binary classification of driving behaviors. Their model was tested on the UAH-DriveSet dataset for the identification of three classes and two classes of driving behavior, attaining an F1-score of 99.49% and 99.34%, respectively, thus outperforming prior state-of-the-art techniques.

III. PROPOSED METHODOLOGY

A. Problem Statement

Fast development in ITS has made it possible to integrate wireless communications between a vehicle and other vehicles (V2) and between vehicles and infrastructures, (V2I). This will enable the sharing of very vital information that shall enhance the management of traffic, its safety, and efficiency. Key challenges that remain for these improvements are requisite with an accurate classification of driver behaviour, which shall be critical for developing adaptive and responsive ITS solutions.

The challenge to be addressed in this paper is how to classify driver behaviours only using artificial intelligence techniques with data sourced from a mobile application. Precisely, the focus will be on the training of a neural network model especially the Long Short-Term Memory kind of networks for the analysis and classification of driving patterns. This task, however, is quite well-suited to LSTM networks due to their ability to learn temporal dependencies and sequential patterns of data, which are inherently found in driving behaviours.

This work is focused on the development of a driver classification algorithm that makes use of a resilient LSTM model in order to predict the drivers behaviour for ITS. It thus has huge potential for improving strategies that govern traffic management, lowering accident rates, and ensuring overall safer and more efficient systems of transportation.

B. Driving classification based on Long Short-Term Memory

Long Short-Term Memory is an improved version of the recurrent neural network (RNN) designed by Hochreiter & Schmidhuber [2]. LSTM is well suited for sequence prediction tasks and excels at capturing long-term dependencies.

A traditional RNN has a single hidden state that is transmitted over time, which can make it difficult for the network to learn long-term dependencies. LSTMs solve this problem by introducing a memory cell, which is a container that can store information for an extended period of time [5] . LSTM networks are capable of learning long-term dependencies in sequential data, which makes them suitable for tasks such as language translation, speech recognition, and time series prediction [5]. LSTMs can also be used in combination with other neural network architectures such as

Convolutional Neural Networks (CNNs) for image and video analysis [5]. Figure 1 illustrates an LSTM cell:
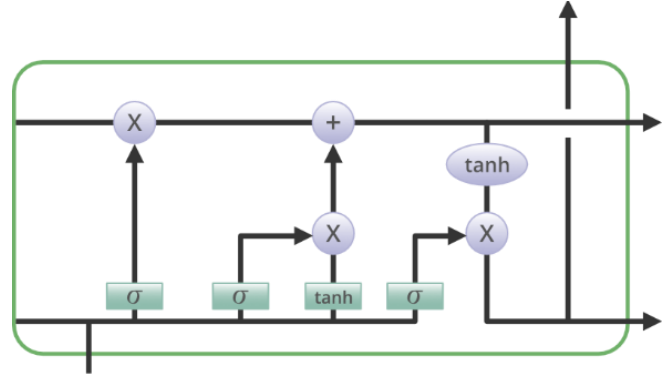

Figure 1 - Diagram of an LSTM cell

An LSTM includes a series of memory cells, blocks responsible for storing and processing information over time. Every LSTM cell holds three gates: the Input Gate, the Forget Gate, and the Output Gate. Here are the equations *(1), (2) and (3)* for the respective Gates:

1) *Input Gate Equation*

$$ft = (\ W_f\ [h_{t-1,}\ x_t] + b_f\ ) \qquad (1)$$

2) *Forget Gate Equation*

$$i_t = (\ W_i\ [h_{t-1,}\ x_t] + b_i\ )$$
$$\hat{C}_t = tanh\ (\ W_c\ [h_{t-1,}\ x_t] + b_c\ ) \qquad (2)$$

3) *Output Gate Equation*

$$o_t = (\ W_o\ [h_{t-1,}\ x_t] + b_o\ ) \qquad (3)$$

C. Long Short-Term Memory Networks Combined with Convolutional Neural Networks

Convolutional Neural Networks, particularly Conv1D, are computationally less demanding, and thus faster to train and run in comparison with traditional LSTMs, especially using Conv1D.

Such architecture is very proficient in extracting relevant patterns from subsequences, which has a positive impact on the results obtained. Because CNN provides the opportunity to apply pooling layers, the dimensions of data are reduced, and this reduction reduces model complexity, helps avoid probable overfitting. However, the major benefit in usage that stands out would be this architecture's ability to combine with others, such as LSTMs, in improving results and helping with scalability. Figure 2 illustrates an ConV1D architecture:
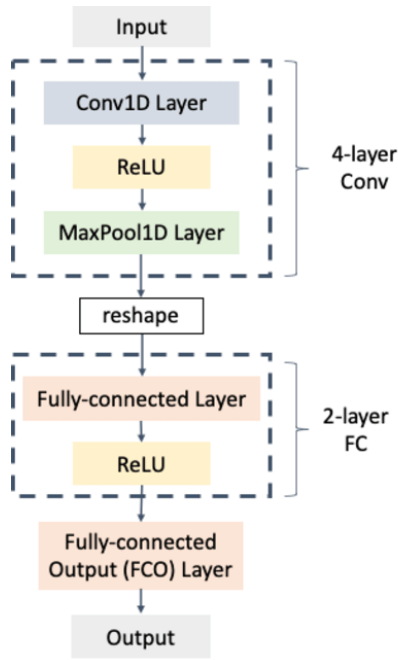
Figure 2 - Diagram of a 1D CNN Implementation using Max Pooling

The effectiveness of 1D Conv depends mainly on the input data structure since in cases where the data does not have a clear temporal pattern or sequences, 1D Conv will not produce accurate results.

D. Data Processing

The data processing consisted in the identification and categorization of different driving manoeuvres:

- Sudden acceleration and deceleration

- Turns and lane changes

- Abrupt stops and start

This was performed by the use of six sensors detecting both the accelerometers and gyroscopes along the axes of X, Y, and Z. In general, accelerometers detected linear acceleration, gyroscopes detected angular velocity, and the measurement was taken along the same axes. After data collection, a preprocessing step was applied to separate positive and negative values. This was to allow the model to have a maximum probability of not being affected by potentially negative numbers. As a result, two columns were created: one containing only positive values and another containing only negative values collected by the respective sensors.

Subsequently, these two columns were organized into a 2D array consisting of a total of 12 elements: 6 positive and 6 corresponding negative elements derived from the processed sensor data. Figure 3 illustrates the data processing process:
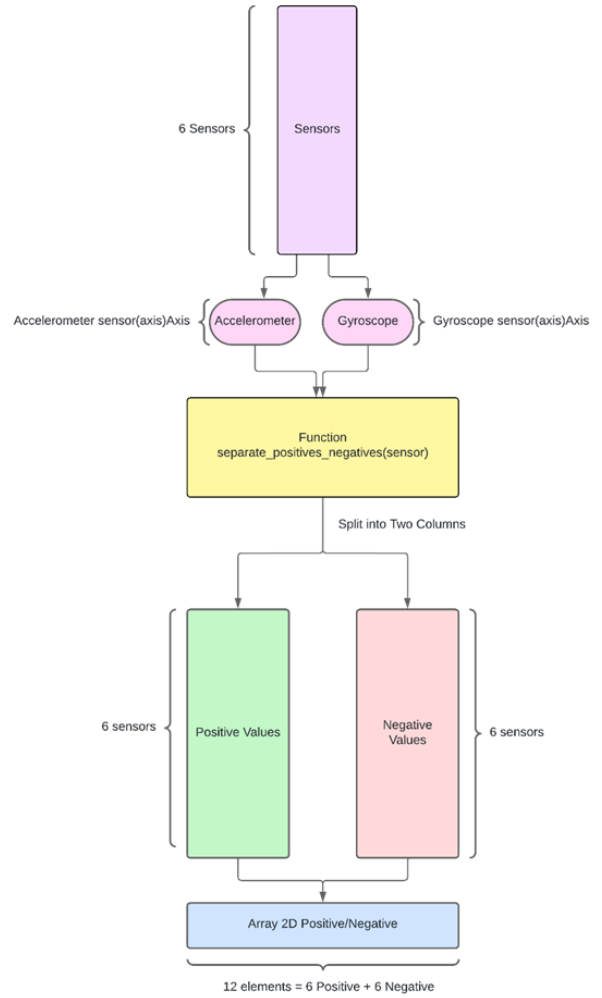

Figure 3 - Diagram of the Data Processing Process

E. Data Classification

The process starts by inputting two parameters into the function **y_classification(data, threshold)**: data and threshold. The intention of the **y_classification** function is to return a Boolean value classifying the data in relation to **1**, meaning aggressive behaviour, and **0** for non-aggressive behaviour. It initializes an **output_vector** that will hold these classifications.

This will involve looping through all columns of the dataset (all 12) and compute the maximum value of each column, done using the numpy function **np.max(data[:,col])**. After getting the maximum value of every column, **threshold_pos** was computed by multiplying this maximum value by a threshold parameter passed as input to **max_value * threshold**.

The classification of the data is then carried out as follows:

- If the data value is greater than or equal to the **threshold_pos**, it is classified as 1 (**aggressive**).

- If the data value is less than **threshold_pos**, it is classified as 0 (**non-aggressive**).

Finally, the vector, initialized with the result of the classification, is returned with this data classified. Figure 4 depicts the data classification process:
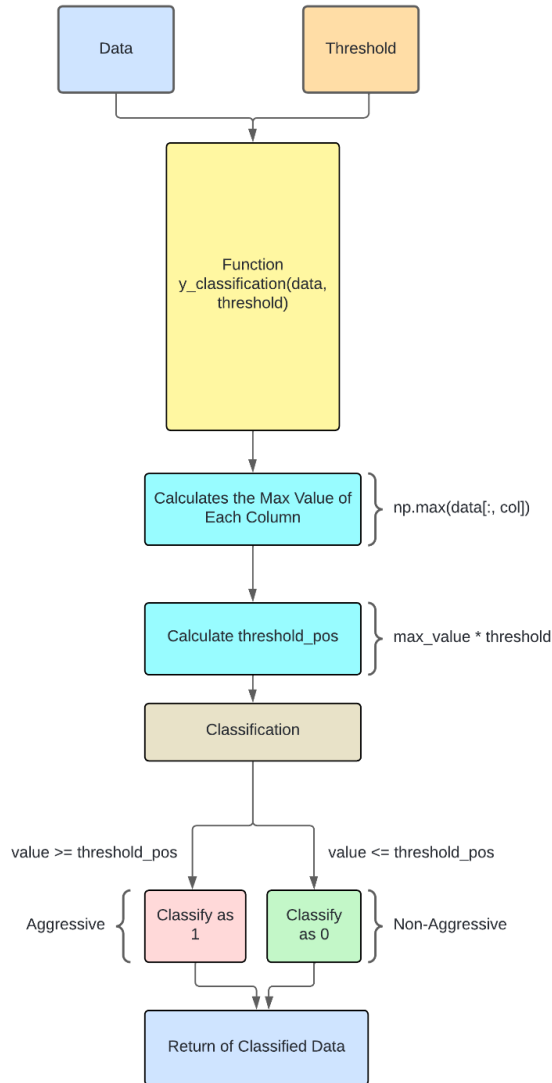

Figure 4 - Diagram of the Data Classification Process

F. Data Normalization

The data was normalized for improving the performance of the model. This normalization involved scaling, that is the adjustment of the values of each variable to some common range, say **[0, Max Value of each Sensor Set]**.

The data normalization process starts with the application of the **'max_of_vectors'** function, which can have two variations:

- For the case of an accelerometer device, the input parameters are: **turnRightX, turnLeftX, accelY, breakY, positiveZ, negativeZ**.

- For the gyroscope, the input parameters are**: gyrPositiveX, turnLeftX, accelY, breakY, positiveZ, negativeZ.**

This function concatenates all columns of input into one vector and returns the maximum value of this vector. Figure 5 shows the application of the 'max_of_vectors' function to the accelerometer:
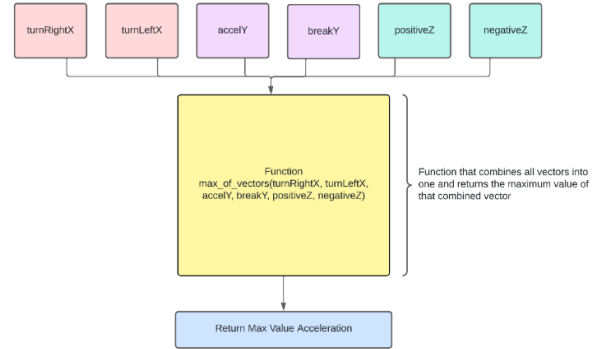

Figure 5 - Diagram of the Max Of Vectors Function Applied to the Accelerometer

Figure 6 demonstrates the application of the 'max_of_vectors' function to the gyroscope:
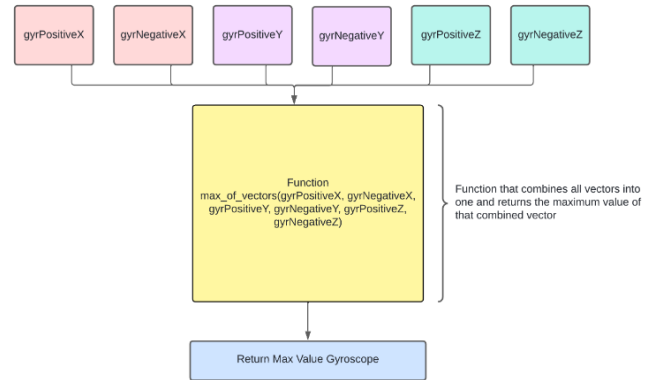

Figure 6 - Diagram of the Max Of Vectors Function Applied to the Gyroscope

The result will be the maximum value of the accelerometer that is applied to normalize function as the input parameter. Figure 7 represents the data normalization process:
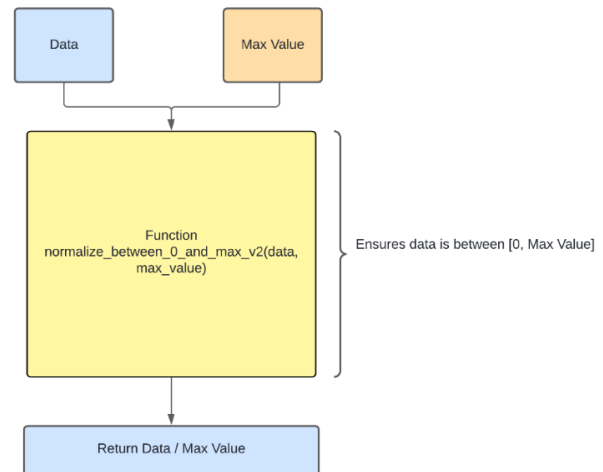

Figure 7 - Diagram of the Data Normalization Process

G. Split Data into Training and Test Sets

The model is tested on part of the data divided for training and testing. This was done in a ratio as follows:

- 80% : Training Data
- 20% : Testing Data

The process of dividing the data into training and testing sets used the function **split_train_test(data, test_size=0.2)**. The input parameters to the function are the data and the size of the test data sequence. The size parameter impacts directly on the size of the training sequence.

Specifically, **test_size = 0.2** means **20%** of the data will be used for testing the model, while the remaining **80%** stays in the train sequence. This is an equal and fair split to both so that training can be effectively done based on it and performance evaluation can also be reliable. Figure 8 outlines the data separation process:
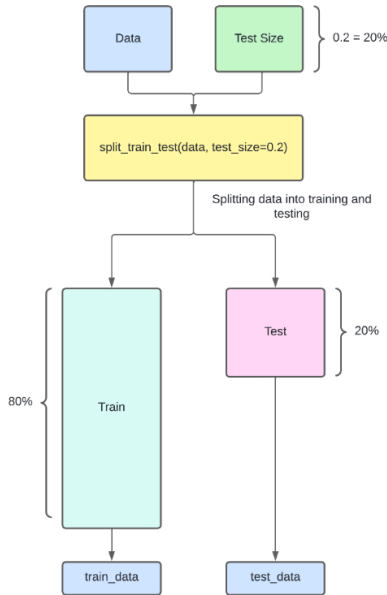

Figure 8 - Diagram of the Data Separation Process

H. Data Representation

To a better analyse of the recorder manoeuvres, we use the Folium Library to create an interactive map. A marker cluster was added to the map to visually group nearby event.

Markers were then added for each recorder position, differentiated by colors to represent different type of manoeuvres. Figure 9 presents a plot of the dataset manoeuvres:
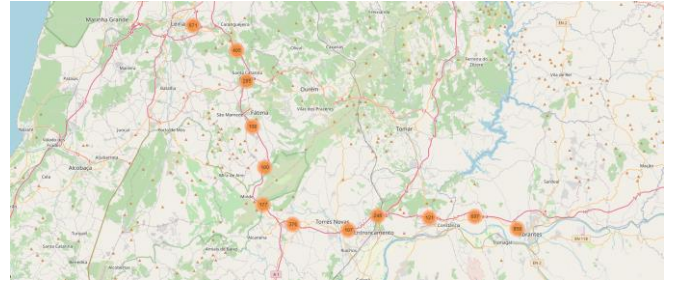

Figure 9 - Plot of Dataset Manoeuvres

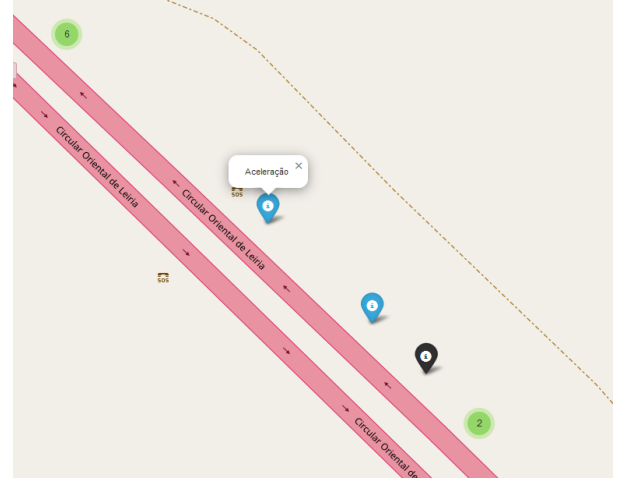Figure 10 provides a zoomed-in plot of the dataset maneuvers:

Figure 10 - Plot of Zoomed Dataset Manoeuvres

I. Model Construction

The model adopted in the present study is a hybrid architecture that presents both convolutional neural networks with one dimension (Conv1D) and long short-term memory networks, which is designed for the classifier of driving behaviours based on sensor data.

1) Model architecture

a) Conv1D Layer
- Filters: 64
- Kernel Size: 1
- Activation: ReLU
- Input Shape: (number of time steps, number of features)

Where Conv1D is the convolutional layer that helps to extract local features from time-series input data, using 64 such filters of size 1 captures fine-grained patterns in this data.

b) Batch Normalization :
This layer helps to normalize the output of the Conv1D layer in order to increase training stability and speed.

c) Dropout Layer:
- Dropout Rate: 0.5

Dropout is there to prevent overfitting, as it randomly sets half of the Conv1D layer output units to zero during training.

d) Primary LSTM Layer:
- Units: 256
- Return Sequences: True

The first LSTM layer captures the temporal dependencies within the data and returns sequences for processing by the next LSTM layer.

e) Dropout Layer
- Dropout Rate: 0.2

For the prevention of overfitting, dropout is applied on the output of the first LSTM layer.

f) LSTM Layer (Middle):
- Units: 128

The second layer of the LSTM network further deals with the temporal dependencies existing in data, reducing the output to a fixed-size vector.

g) Dropout Layer:
- Dropout Rate: 0.2

To prevent overfitting, the output of the second LSTM layer is passed to a Dropout layer.

h) Dense Layer:
- Units: 12
- Activation: Sigmoid

The Dense layer using 12 units and the activation function sigmoid give the final classification scores for each class.

2) Model Compilation

The model was built using the following parameters:

- Optimizer: **Adam**
- Loss function: **Binary Crossentropy**
- Metrics: **Accuracy**

The Adam optimizer employs high-level adaptive learning rates and demonstrates efficiency, which may alleviate the typical problems of training deep neural networks [1]. The binary crossentropy loss is selected as the criteria based on which the structure is optimized, and the model is evaluated through validation using Accuracy. This makes the hybrid Conv1D-LSTM model able to extract the strengths of these convolutional and recurrent layers to derive spatial and temporal features from sensor data, making it good at performance in driving behaviour classification.

IV. EXPERIMENTS

In our experiments we focus on comparing several advanced models, including Stacked LSTM, ConvLSTM, and BiLSTM. The selection of the models used in this research study has been based on the effectiveness associated with handling temporal and spatial dependencies.

The dataset for this work is designed to record all types of sensor data from mobile devices. It has many contents and holds two types of data, which include JSON and CSV, giving flexible data and analysis.

A. DataSet

The dataset used in this work is designed to record all types of sensor data from mobile devices. It comprises several contents and supports two types of data, JSON and CSV, and offers versatile data handling and analysis.

1) CSV Structure

a) Label
- Name identifier for the dataset.

b) CreatedAt
- The exact time and date that the dataset was created.

c) CaptureSpeed
- Data capture frequency in milliseconds.

d) GravityFilter
- An option to, if on, remove gravitational effect from accelerometer values.

e) AverageFilter
- Smoothes sensor values.

f) ElapseTimeInSeconds
- Duration in seconds when data has been captured.

g) Device Information
- AndroidID
  - A Unique Identifier for the device.
- Name
  - User-defined device name.
- Manufacturer
  - Device brand name.
- Brand
  - Brand of the device.
- Model
  - Device model.
- Accelerometer
  - Includes maximum range, sensor name, minimum delay, and readings on X, Y, and Z axes.
- Gyroscope
  - Includes maximum range, sensor name, minimum delay, and readings on X, Y, and Z axes.
- Location Data
  - Latitude, longitude, and speed in km/h.

h) Timestamp
- The date and time for every entry of data.

B. Training

The selection of accuracy as the primary metric for our Long Short-Term Memory LSTM model is grounded in its relevance to our driving behavior classification problem and the strengths of LSTM networks.

First, accuracy is the most direct and understandable metric for model correctly identifying driving behaviors. This measures the percentage of accurate predictions compared to total predictions done by the model, therefore it is a clear and transparent indicator for evaluating how well our model performs.

Additionally, accuracy is particularly appropriate for our driving behavior data because it gives a straightforward summary index that describes the testing performance of model. While a few other metrics can be swayed by certain issues like class imbalance or outliers, accuracy gives an unbiased view on how well the model identifies both common and rare driving behaviors.

The training of the Conv1D-LSTM model was further augmented by deploying callbacks to provide the best model performance and to protect against overfitting.

With the *ModelCheckpoint* callback, each training epoch if the val_loss was lower than the previous epoch, the model saved as the best model. In this way, only the model having the lowest validation loss will be used for a final evaluation and deployment.

We also use an early stopping callback, which paused the training process when validation loss did not improve for several epochs. This method avoided overfitting and saved some extra time for training.

The model was trained using the fit method passing in the training (train and y_train) and validates it with validation (test and y_test). This was set up to run for 30 epochs while running a batch size of 256, though, technically, this could stop earlier due to the early stopping callback.
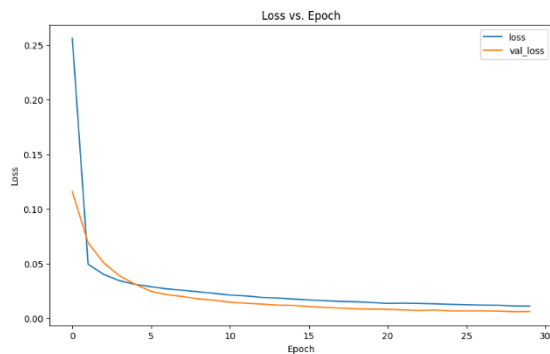
C. Performance Evaluation

Figure 11 - Loss and Val_Loss Throughout the Epochs

As shown in Figure 11, the Loss and validation Loss during training converged within 30 epochs. From this plot, it is clear that both training Loss and validation Loss drop sharply in the early epochs before stabilizing. This basically means that most models learn well with respect to the data presented to them for training and generalize quite well on their validation sets as well.
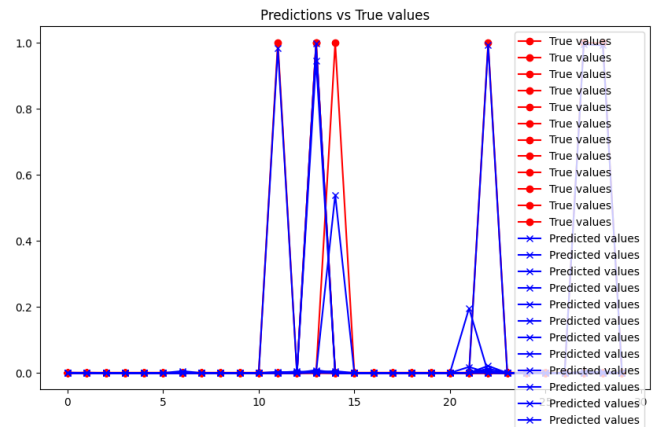

Figure 12 - Model Performance Graph

This plot in Figure 12 represents part of the test data predicted by the model against true values, in order to show how close the models predictions were to the real values and which models differed.

D. Results Comparison

Evaluation of the implemented models was supported by several metrics and visualizations. Hence, the major aspects of their evaluation include Accuracy over training epochs, with other relevant metrics such as accuracy, precision, recall, F1 score, Hamming loss, and Jaccard score. Table 1 compares the performance metrics of the three models built:

Table 1 - Metrics Comparison Between Models

Metrics	Models		
	Proposed ConvLSTM	*BiLSTM*	*Stacked LSTM*
Accuracy	97.71%	96.06%	96.38%
Precision	95.40%	84.66%	95.00%
Recall	93.02%	87.56%	85.65%
F1 Score	93.96%	81.66%	89.29%
Hamming Loss	0.20%	0.33%	0.32%
Jaccard Score	88.95%	72.69%	81.95%

In the Table 1, can compare the different results of the metrics for the models built. As we can observe, all models exhibit high accuracy, with the proposed model reaching an accuracy rate of 97.71%. This demonstrates that in most instances, the models are able to correctly classify a driving behaviour. The precision and recall values are also high, with Proposed ConvLSTM and Stacked LSTM particularly strong performance, suggesting that these models are effective in identifying relevant driving behaviours without many false positives or negatives. Definitely, the balanced performance of the models, with the present proposal ConvLSTM, can be confirmed with this F1 score, which is found as a harmonic mean of Precision and Recall, leading at 93.96%.

The low values that Hamming loss acquires are a few fractions of incorrectly predicted labels. Proposed ConvLSTM has as low as 0.20% of Hamming loss.

High Jaccard scores indicate that the model has strong overlap between the predicted and real driving behaviours, where Proposed ConvLSTM leads again at 88.95%. Overall, the best performance with respect to most of the metrics in this study is the proposed model, ConvLSTM, which can be interpreted as fairly superior with due respect to capturing and predicting driving behaviours accurately.

V. Conclusion

In this work, we developed and compared three LSTM-based models for driving classification problem: Stacked LSTM, ConvLSTM, and Bidirectional LSTM. This study is carried out to find the efficiency of each architecture in capturing temporal dynamics from driving data and their robustness across different driving conditions.

Very importantly, our experiments have shown that each model has its strengths. The Stacked LSTM model had high learning capacity for complex temporal patterns due to the presence of multiple layers of LSTMs, thus performing well in tasks involving intricate sequential dependencies. Then, using convolutional layers, the ConvLSTM model was able to capture features in both the spatial and time domains, outperforming others in scenarios that required detailed analysis in the space-time dimension. The Bidirectional LSTM performed very well at understanding context from both past and future sequences and turned in superior performance in tasks that require comprehensive sequence context.

In this work, we have contrasted these LSTM architectures with respect to their peculiar advantages and limitations in the context of autonomous driving. More importantly, enhanced attention mechanisms ensure AI-driven decisions are made transparently and become more understandable.

Although the models were extremely promising, there are a number of topics that clearly open up avenues for further investigation. Future research shall be focused on optimization of these models for real-time applications, where scalability remains challenging. Expanding training datasets to hold more varied driving environments will increase model robustness. Further development of methods for interpretability is also required to make AI systems able to provide clear and human-understandable explanations for their decisions.

References

[1] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," arXiv preprint arXiv:1412.6980, 2014.

[2] LSTM Full Form - Long Short-Term Memory - GeeksforGeeks. (n.d.). Retrieved July 17, 2024, from https://www.geeksforgeeks.org/lstm-full-form-long-short-term-memory/

[3] S. Bouhsissin, N. Sael and F. Benabbou, "Driver Behaviour Classification: A Systematic Literature Review," in IEEE Access, vol. 11, pp. 14128-14153, 2023.

[4] K. Saleh, M. Hossny and S. Nahavandi, "Driving behaviour classification based on sensor data fusion using LSTM recurrent neural networks," 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC), Yokohama, Japan, 2017, pp. 1-6

[5] What is LSTM - Long Short Term Memory? - GeeksforGeeks. (n.d.). Retrieved July 17, 2024, from https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/

[6] Darsono, A. M., Mat Yazi, N. H., Ja'Afar, A. S., Othman, M. A., & Ahmad, M. I. (2024). Utilizing LSTM Networks for the Prediction of Driver Behaviour. Przeglad Elektrotechniczny, 2024(4), 182–185. https://doi.org/10.15199/48.2024.04.34

[7] Deo, N., & Trivedi, M. M. (n.d.). Multi-Modal Trajectory Prediction of Surrounding Vehicles with Maneuver based LSTMs.

[8] M. A. Khodairy and G. Abosamra, "Driving Behavior Classification Based on Oversampled Signals of Smartphone Embedded Sensors Using an Optimized Stacked-LSTM Neural Networks," in IEEE Access, vol. 9, pp. 4957-4972, 2021