applied sciences

MDPI

Article

A Multimodal Recurrent Model for Driver Distraction Detection

Marcel Ciesla * and Gerald Ostermayer

Research Group Networks and Mobility, University of Applied Sciences Upper Austria, Softwarepark 11, 4232 Hagenberg, Austria; gerald.ostermayer@fh-hagenberg.at

* Correspondence: marcel.ciesla@fh-hagenberg.at or marcel.ciesla1707@gmail.com

Abstract: Distracted driving is a significant threat to road safety, causing numerous accidents every year. Driver distraction detection systems offer a promising solution by alerting the driver to refocus on the primary driving task. Even with increasing vehicle automation, human drivers must remain alert, especially in partially automated vehicles where they may need to take control in critical situations. In this work, an AI-based distraction detection model is developed that focuses on improving classification performance using a long short-term memory (LSTM) network. Unlike traditional approaches that evaluate individual frames independently, the LSTM network captures temporal dependencies across multiple time steps. In addition, this study investigated the integration of vehicle sensor data and an inertial measurement unit (IMU) to further improve detection accuracy. The results show that the recurrent LSTM network significantly improved the average F1 score from 71.3% to 87.0% compared to a traditional vision-based approach using a single image convolutional neural network (CNN). Incorporating sensor data further increased the score to 90.1%. These results highlight the benefits of integrating temporal dependencies and multimodal inputs and demonstrate the potential for more effective driver distraction detection systems that can improve road safety.

Keywords: driver distraction detection; driver monitoring; advanced driver assistance systems; AI-based passenger monitoring

check for updates

Citation: Ciesla, M.; Ostermayer, G. A Multimodal Recurrent Model for Driver Distraction Detection. *Appl. Sci.* **2024**, *14*, 8935. https://doi.org/10.3390/app14198935

Academic Editor: Pedro Couto

Received: 20 August 2024
Revised: 16 September 2024
Accepted: 26 September 2024
Published: 4 October 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Driving is a complex task that requires full attention, including constant awareness of the environment, monitoring of the road, and readiness to react to unexpected events [1,2]. Driver distraction compromises these essential factors and reduces the ability to drive safely. The human brain struggles to manage multiple tasks simultaneously, particularly when the tasks are similar or require continuous focus, which can negatively affect task performance [1].

Distractions are categorized into four types based on their source: visual, auditory, manual/physical, and cognitive [3]. Visual distractions occur when the driver's eyes are taken off the road, such as checking a phone. Auditory distractions are sounds that take attention away from driving, such as a ringing phone, although some sounds, such as safety notifications, can be beneficial (also called positive distractions). Manual/physical distractions occur when drivers use their hands for non-driving tasks, such as eating or using a smartphone. Cognitive distractions occur when the driver's mind strays from the driving task and competes with the mental focus needed to drive safely.

Despite significant efforts in recent decades to improve road safety, the global incidence of road traffic crashes continues to claim the lives of at least one million people every year [4]. Among the various causes of road crashes, distracted driving is one of the most common, along with drunk driving. In the European Union, driver distraction is estimated to account for between 10% and 30% of all road accidents [5]. In particular, the use of a smartphone while driving is widespread, especially among younger drivers. As a result, many countries have introduced legislation that imposes fines for using a smartphone while driving [2]. Although other distractions may not be illegal, they still affect driving

performance, such as interacting with the infotainment system, talking to passengers, and eating or drinking while driving. Depending on the nature of the distraction, these activities can have different effects on the driver's perception and significantly increase reaction times, thereby increasing the risk of a collision.

Various strategies can be used to reduce the prevalence of distracted driving. In addition to enforcing stricter penalties and raising awareness through public campaigns and driver training, implementing countermeasures directly in vehicles is also an effective approach [2]. Modern vehicles are equipped with advanced driver assistance systems (ADASs) designed to assist the driver in performing the driving task. Systems such as emergency braking assistants and lane departure warnings significantly reduce the risk of road accidents. The current trend is even toward the development of partially or fully automated vehicles, where control of the vehicle is completely transferred to the system. However, until fully automated vehicles are on our roads, the human driver still acts as a supervisor of the system [6]. In partially automated systems, even when the vehicle is in control, the driver must remain alert and ready to intervene quickly in dangerous situations that the vehicle cannot handle on its own. However, the increased comfort of self-driving situations may tempt drivers to engage in distracting activities, which may impair their ability to intervene in a timely manner. A promising solution to reduce the prevalence of distraction on the road is the implementation of driver distraction detection systems [2]. These systems, which use technologies such as camera-based monitoring, can detect and warn drivers when they are distracted. By reducing the frequency of distracting activities, drivers are encouraged to remain focused on their driving tasks. Consequently, in partially automated vehicles, this approach keeps the driver actively involved in the driving task, enabling them to safely take control in critical situations.

To mitigate the risk and reduce the prevalence of distracted driving, the implementation of driver distraction detection systems has emerged as a promising solution. These systems are designed to detect distractions and provide timely warnings to the driver. Various AI-based approaches have been explored in research and industry to detect driver distraction, including vision-based methods that analyze driver camera images, sensor-based techniques that use sensor data, and multimodal approaches that combine both image and sensor data [7]. Vision-based methods, such as those used in the State Farm Distracted Driver Detection competition, rely on camera images to classify distractions [8]. Different CNN architectures such as *DenseNet-201* [9], *ResNet-50* [10], and *InceptionV3* [11] have been tested and have achieved high levels of accuracy on datasets such as the State Farm dataset and the AUC dataset [12]. Hybrid models combining CNNs with recurrent neural networks (RNNs), especially LSTMs, have shown improved performance, especially when considering temporal dependencies [12–14]. Sensor-based methods most often use data from non-intrusive sensors such as IMUs and vehicle dynamics data to detect distractions. Studies have shown that LSTM networks, especially those enhanced with attention mechanisms, can effectively classify distractions based on sensor data with high accuracy [15]. Intrusive sensors, while accurate, are less practical for real-world applications due to their impact on the driver [6]. Multimodal approaches that combine visual and sensor data have been shown to improve detection accuracy. For example, combining CNN-based visual analysis with LSTM-based sensor data analysis significantly improves classification performance [7,16]. This fusion of data types allows for more robust and comprehensive detection systems.

Previous research on driver distraction detection has predominantly relied on a vision-based approach, using camera images of drivers as the sole input. These methods typically analyze images on a frame-by-frame basis, without accounting for temporal dependencies between frames. The innovative aspect of this project lies in its investigation of whether incorporating temporal dependencies across multiple time steps can enhance detection accuracy. This introduces a novel approach compared to classical, non-recurrent vision-based models by leveraging sequential information to improve detection confidence. In addition, the project makes a significant contribution by exploring the benefits of multi-

modal input, specifically integrating both image and sensor data into the detection model. This integration of vehicle dynamics and steering wheel movement data is hypothesized to offer a new dimension to distraction detection, providing greater accuracy than image data alone. In summary, this paper presents a novel approach by incorporating both temporal components and multimodal data into a vision-based driver distraction detection system to assess whether this combination leads to improved predictive performance.

2. Materials and Methods

A key advance of this work over previous research is the use of a recurrent machine learning model that processes multimodal input. Unlike approaches that evaluate a single moment in time, such as analyzing a single image, this model incorporates multiple time steps in the decision-making process, aiming for more accurate detection. In addition, this model goes beyond a focus on image data alone by integrating sensor data to create a comprehensive multimodal framework. Specifically, it incorporates vehicle data from the OBD2 (on-board diagnostics) interface and additional data from an inertial measurement unit (IMU) mounted on the steering wheel, which captures detailed information on vehicle dynamics and steering wheel movements. The primary objective of this project was to evaluate whether the combined use of a time-dependent model and the fusion of image and sensor data can improve classification performance compared to a purely vision-based approach.

2.1. Data Collection

During this project, a custom dataset was collected to train the models. The dataset was collected from trips made by six different drivers along a predetermined 23 km route in a rural area, characterized by numerous curves and including sections on both main roads and local roads. Each trip lasted between 25 and 30 min, depending on traffic conditions. During these trips, drivers were randomly instructed to perform specific distracting tasks to simulate real-world distracted driving scenarios. Data collection involved using a camera to record the drivers and a microcontroller equipped with an IMU (mounted on the steering wheel) to capture acceleration and gyroscope data, providing insights into vehicle dynamics and steering movements.

However, it is important to note that the collected data are insufficient to train a robust driver distraction detection model for real-world vehicle applications. Training such a model would require a significantly larger and more diverse dataset, representing different forms of distracted driving across a wider range of individuals, vehicles, recording conditions, and driving scenarios. While the current dataset is not sufficient for real-world model training, it does allow the evaluation of different technologies, in particular comparing the effectiveness of a vision-based approach enhanced by a recurrent model and additional sensor data. To ensure consistency and control of the dataset, the same vehicle and route were used for each recording, and efforts were made to maintain similar lighting conditions.

2.1.1. Recording Setup

The recording setup was centrally managed from a notebook, which acted as a control center during the drives. A Python application was developed to handle communication with the data sources and to perform initial preprocessing tasks, such as decoding OBD2 data. Data from each recording session was organized in a timestamped folder on the notebook.

The setup included three key data sources:

- Camera: A *Logitech C920 PRO HD* webcam was mounted on the passenger side, facing the driver. It captured 30 frames per second at 1920 × 1080 resolution, which was later downscaled to 224 × 224 pixels for model training. Each image was timestamped and stored in the recording folder.

- IMU (steering wheel): A microcontroller (*NodeMCU Lua Amica Module V2 ESP8266*) equipped with a 9-axis IMU (*BlueDot BNO055*) was attached to the steering wheel to record vehicle dynamics and steering movements. IMU data, including acceleration, angular velocity, and magnetic field strength, was transmitted to the notebook via MQTT protocol at 40 samples per second. The data were stored in JSON files with regular checkpoints to prevent loss.
- OBD2 Interface: A Bluetooth-enabled OBD2 adapter (*Veepeak OBDCheck VP11*) was used to capture additional vehicle sensor data. The laptop established a Bluetooth connection to continuously request and record parameters such as vehicle speed, engine speed, and throttle position at a frequency of 5 Hz. These data were also stored in JSON files alongside the IMU data.

2.1.2. Distraction Tasks

As mentioned above, all drivers were explicitly instructed to perform specific distracting activities during the recording sessions. The distracting tasks included

- Drinking: Each driver was given a water bottle and instructed to drink from it during the ride without being asked.
- Looking to the passenger: This task introduced visual and cognitive distraction by instructing drivers to direct their attention to the passenger side. To achieve this, a set of A4 slips of paper was prepared, each containing different AI-generated random texts. During the drive, one of these slips was repeatedly held up and the drivers were asked to read the text aloud.
- Operating the infotainment system: A number of activities involving the operation of the infotainment system were carried out. As most of the drivers were unfamiliar with the car, they were given a brief introduction to the vehicle before the recording. The infotainment tasks included entering a destination in the navigation system, changing the radio station, and changing the current driving mode (e.g., from efficiency mode to comfort mode).
- Reaching behind: This task involved forcing a distraction in which the driver had to reach behind. Several packs of tissues were placed in the passenger seat back pocket and each driver was asked to reach for one of the packs and hand it to the passenger several times while driving.

2.2. Data Preprocessing

Several preprocessing and feature engineering steps were required to convert the raw data from the recordings into a format suitable for training a machine learning model.

2.2.1. Temporal Synchronization

The first critical step was to synchronize the different data sources (camera, OBD2, IMU) for each recording due to their asynchronous nature. Temporal synchronization was achieved by time-aligning each image with the corresponding sensor data. The camera data, recorded at 30 frames per second, were aligned with the latest IMU and OBD2 data, which had different sampling rates (40 samples per second for IMU and 5 samples per second for OBD2). This process resulted in some IMU data being dropped and each OBD2 data vector being associated with multiple frames. The output of this synchronization is a *Pandas DataFrame*, where each row corresponded to a specific recording time, with the first column containing the path to the camera image and the remaining columns containing the sensor data features.

2.2.2. Labeling

The labeling process aimed to capture the driver's state at each moment of the recording. The distraction classes *drinking*, *looking_to_passenger*, *infotainment*, and *reaching_behind* were defined based on the specific instructions given during the recordings. *Normal_driving* was defined as any time the driver was alert and driving safely with both hands on the

wheel. For each recording, a separate CSV file was created to specify the time ranges for each driver state over time. These time ranges were then used to label the appropriate class for each moment in the recording.

To ensure consistent labeling, specific criteria were used to determine the start and end of each distraction. For example, the *drinking* sequence began when the driver released the steering wheel to reach for the bottle and ended when the hand returned to the steering wheel. Similarly, *infotainment* and *reaching_behind* were marked by the release and return of the driver's hand to the steering wheel. The *looking_to_passenger* distraction began when the driver first looked at the note and ended when the sentence was read aloud and attention returned to the road.

Initially, only the distraction periods were labeled, assuming that the rest of the time was *normal_driving*. However, it was found that other "unplanned" distractions such as scratching, sneezing, or adjusting glasses occurred during these periods. These actions were grouped into a *other* class, which was used for statistical analysis but excluded from model training. The labeling process added a *class* column to the data frame, representing the driver's state at each time point. This was further encoded into a numeric *label* column using the *LabelEncoder* from the *scikit-learn* module, mapping each distraction class to a specific numeric value.

2.2.3. Feature Scaling

After temporal synchronization and labeling, the next step was to scale all numerical features, especially the sensor data from the OBD2 interface and IMU. Feature scaling can be critical to the performance of matrix-based machine learning models such as neural networks [17]. To achieve this, the *StandardScaler* from the Python *scikit-learn* module was used, which standardizes features by subtracting the mean and scaling by the standard deviation.

2.2.4. Creating Sequences

Since the project involves the analysis of sequential data, the goal of this step was to generate sequences suitable for training a recurrent neural network, such as an LSTM network. A Python method was developed to generate sequences from the data frame with a specific temporal length and stride (overlap between consecutive sequences). This method organized the data by driver and distraction sequence and divided it into smaller, temporally defined sequences. Each sequence contained a series of images (via relative paths) and corresponding sensor data.

To meet the requirements of recurrent neural networks, which often require a constant sequence length, a pre-padding technique was used. This ensured that all sequences matched the length of the longest sequence by padding shorter sequences with the first value in the sequence. The result was a data frame containing padded sequences of image paths and sensor data, ready for model training.

2.3. Exploratory Data Analysis

An exploratory data analysis (EDA) was performed to investigate the properties and characteristics of the preprocessed dataset. The dataset consists of one data frame, where each time point of a recording corresponds to a row, and another data frame containing sequential data derived from the first one. The following analysis focuses on the first data frame, which summarizes key statistical metrics for the sensor features, as shown in Table 1.

Table 1. Statistical metrics of numerical features.

Feature	Unit	Mean	Std	Min	Max	Median
acc_x	m/s^2	0.05	0.95	−6.20	8.21	0.01
acc_y	m/s^2	0.04	0.79	−3.71	7.42	−0.08
acc_z	m/s^2	0.12	0.70	−5.53	8.18	0.10
rot_x	rad/s	0.03	0.08	−0.66	0.59	0.01
rot_y	rad/s	0.00	0.16	−2.11	2.26	0.00
rot_z	rad/s	0.00	0.38	−6.06	5.33	0.00
speed	km/h	56.13	16.85	0.00	111.00	56.00
acc_pedal_position	%	24.12	8.26	15.00	74.00	24.00
throttle	%	62.64	27.77	14.00	89.00	84.00
engine_coolant_temp	°C	84.18	8.16	42.00	92.00	87.00
engine_load	%	39.60	35.01	0.00	100.00	39.00
engine_speed	rpm	1452.67	286.32	0.00	3677.00	1418.50

The dataset contains 251,271 samples after filtering out invalid data, where each sample consists of a camera image and a vector of numerical sensor data. The features of the data frame include IMU data (e.g., accelerations and angular velocities) and OBD2 sensor data (e.g., vehicle speed and engine parameters). The correlation analysis, visualized in Figure 1, identified a strong positive correlation coefficient (0.93) between *acc_pedal_position* and *engine_load*, leading to the exclusion of the *engine_load* feature from further analysis.

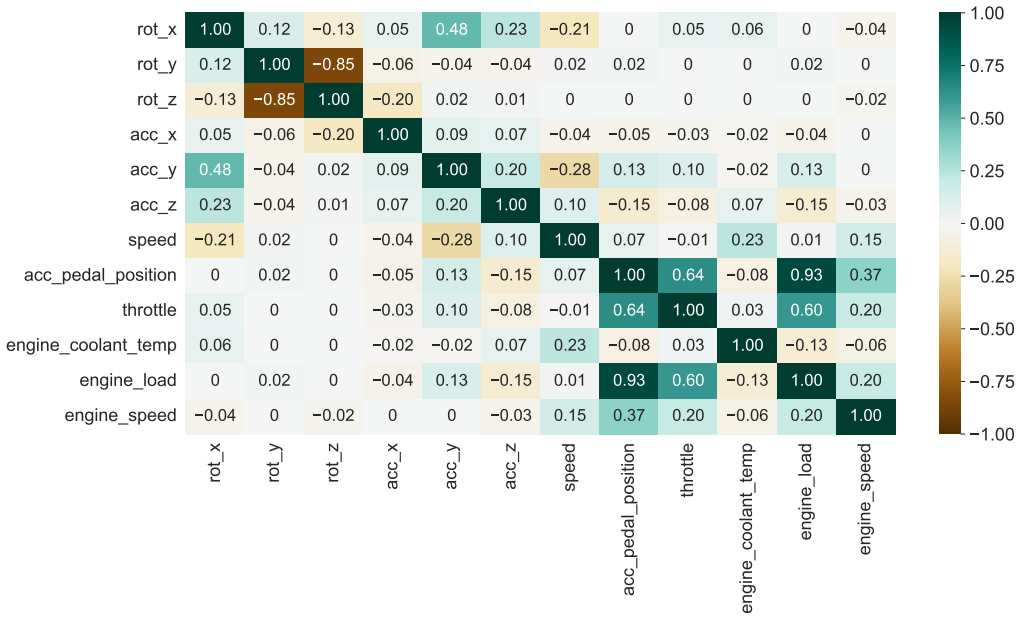


Figure 1. Visualization of a correlation matrix heatmap of the numerical features.

The class distribution across drivers, shown in Figure 2, reveals an unbalanced dataset, with the largest class being *normal_driving* (176,927 samples) and the smallest is *reaching_behind* (7576 samples). This imbalance can adversely affect the training of machine learning models, requiring the use of countermeasures that will be presented later.

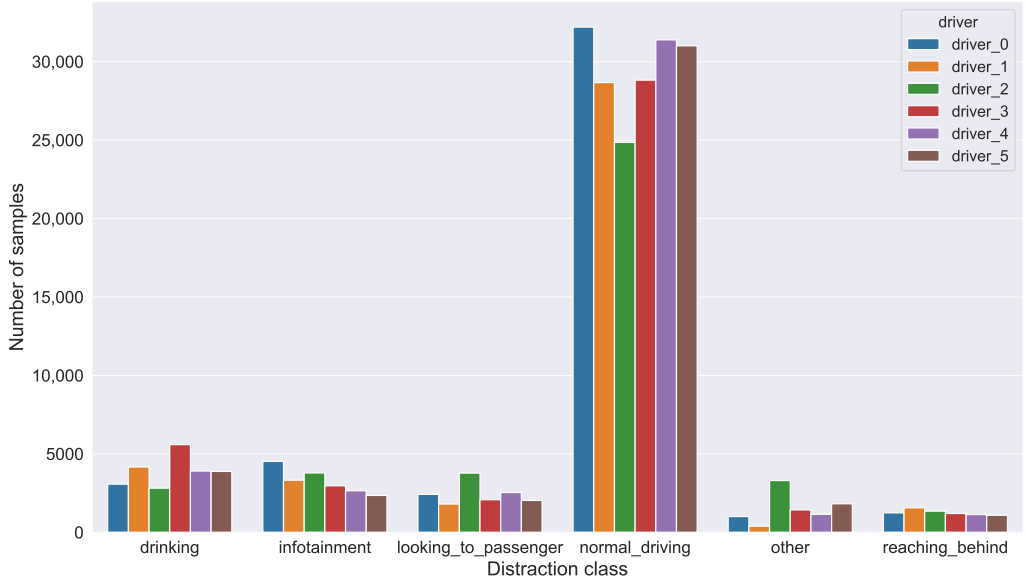


Figure 2. Visualization of the class distribution of the dataset among the drivers.

Distraction durations varied significantly across tasks, as shown in the boxplots in Figure 3. The longest average distraction was *drinking* (15.9 s), while the shortest was *other* (1.98 s). The interquartile ranges indicate that *drinking* and *infotainment* have the greatest variability in duration, while *reaching_behind* has more consistent durations.

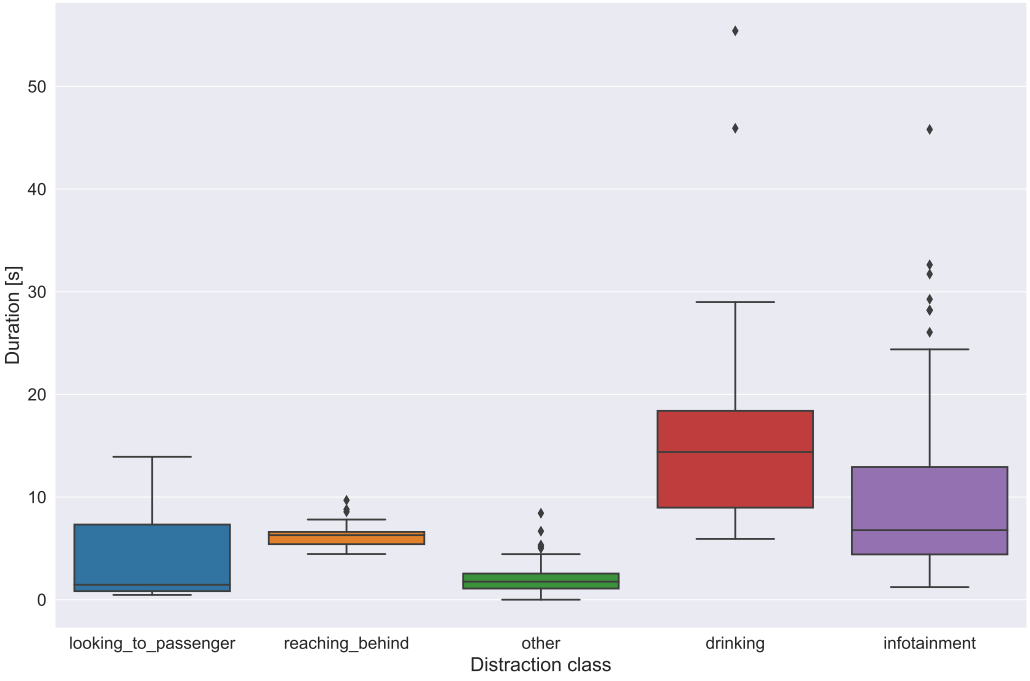


Figure 3. Boxplots with the durations of distracting activities.

Additional plots providing further insights into the distribution of distraction durations are included in Appendix A.

2.4. Model Architecture

Initially, a convolutional neural network was developed for a purely vision-based approach, focusing on image classification. This model was then extended by integrating a long short-term memory network to process image sequences, resulting in a recurrent vision-based model. Finally, the architecture was further extended to include a neural network for processing sensor data, resulting in a multimodal recurrent model capable of handling both image sequences and sensor data for driver distraction detection.

2.4.1. Vision-Based Approach with a CNN

The first phase involved training a CNN model for driver distraction detection using visual information, which served as a baseline for further improvements using LSTM networks and sensor data. The chosen architecture was *VGG16* [18], processing images in the dimensions $224 \times 224 \times 3$. The input passes through five blocks of convolutional layers, each followed by max-pooling layers. The network ends with three fully connected layers, the last of which uses a sigmoid activation function, while the others use *ReLU*. *VGG16*, despite its high parameter count (≈138 million), has shown effectiveness in previous research on driver distraction detection [7,12,13,19].

Figure 4 illustrates the *CNN* model's architecture. After the convolutional layers, a *GlobalAveragePooling2D* layer replaces the typical flattening layer to reduce parameters and improve training efficiency [20]. The model then employs two fully connected layers with *ReLU* activation and dropout layers for regularization [21]. The final layer uses a *softmax* activation function to output probabilities for five distraction classes.

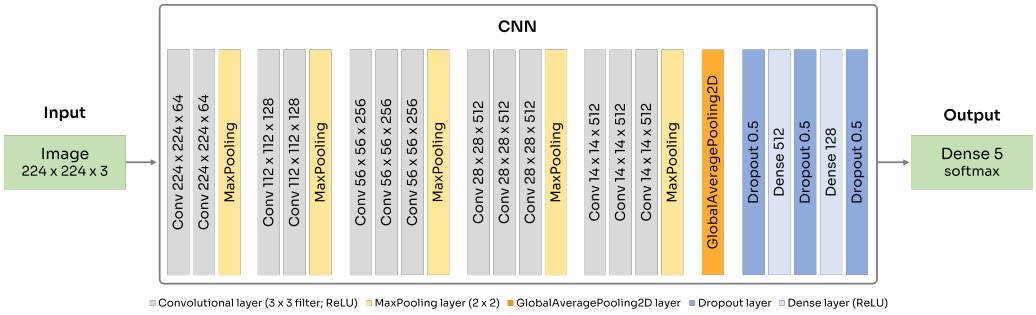


Figure 4. CNN model used for vision-based driver distraction detection.

To counteract overfitting, data augmentation was applied using random translations, rotations, zooms, and contrast adjustments during training. This approach helps the model generalize better by introducing variations in the input images, addressing issues like differing driver seating positions [21].

Given the large number of layers and weights in *VGG16*, transfer learning was employed. *VGG16* weights pre-trained on the ImageNet dataset were used, with only the final convolutional layer fine-tuned for this task, resulting in a model with ≈15 million parameters, of which ≈2.7 million were trainable.

2.4.2. Recurrent Vision-Based Approach with a CNN-LSTM Network

Building on the vision-based approach of the previous section, a recurrent neural network was incorporated to improve the model by considering a sequence of images instead of a single image. This section presents a *CNN-LSTM* model that extends the previous *CNN* with an LSTM network for temporal modeling.

As shown in Figure 5, the model processes an input sequence of images ($seq_len \times 224 \times 224 \times 3$). To maintain consistency for comparison, the same *CNN* from the previous step is reused, wrapped in a *TimeDistributed* layer that applies the *CNN* to each image in the sequence. The resulting sequence of CNN outputs is then fed into two LSTM layers of 128 units each. The LSTM layers are followed by additional fully connected layers with *ReLU* activation, culminating in a *softmax* output layer with 5 neurons to predict the

class probabilities for the different distraction classes. The proposed *CNN-LSTM* model has ≈15.3 million parameters, from which ≈3 million are trainable.

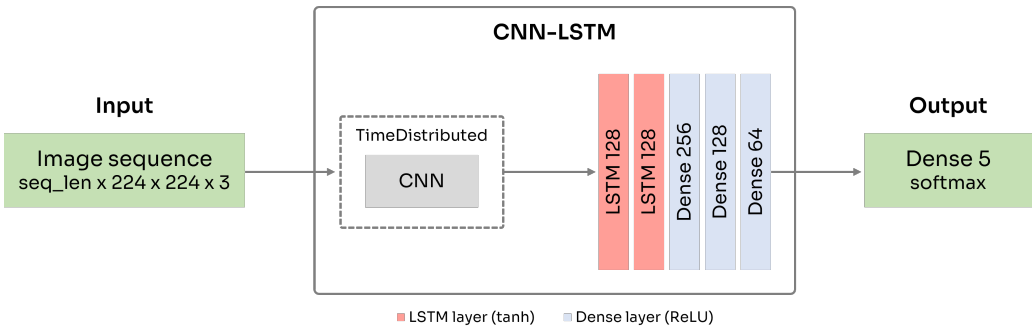


Figure 5. CNN-LSTM model used for recurrent vision-based driver distraction detection.

2.4.3. Multimodal Recurrent Approach with a CNN-LSTM Network and Sensor Data

The final model introduced is a neural network that incorporates both image sequences and sensor data, referred to as a multimodal recurrent approach for driver distraction detection (*CNN-LSTM-S*).

This model extends the configuration of Section 2.4.2 by adding a sensor data input, as shown in Figure 6. The input now contains both an image sequence and a sensor data sequence (*seq_len* × 11). The image sequences are processed by a *TimeDistributed* layer that applies the *CNN* to each image. At the same time, the sensor data are processed by another *TimeDistributed* layer that encapsulates a neural network with two dense layers (32 and 16 neurons). The outputs from these two *TimeDistributed* layers are then merged using a concatenation layer, creating a fused sequence of CNN and sensor data outputs.

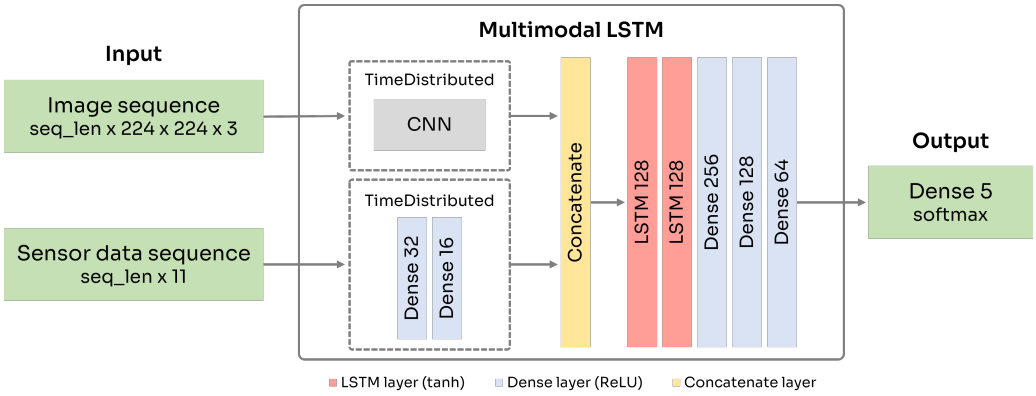


Figure 6. CNN-LSTM-S model used for multimodal recurrent vision-based driver distraction detection.

The fused sequence is passed through two LSTM layers and three dense layers, identical to the previous model, culminating in the output layer that predicts distraction class probabilities. Similar to the *CNN-LSTM* model, the *CNN-LSTM-S* has ≈15.3 million parameters, from which, ≈3 million are trainable.

2.5. Model Evaluation and Training

The models were trained using a university-provided GPU rack equipped with *NVIDIA GeForce RTX 3090* GPUs. *TensorFlow* (with *CUDA* and *cuDNN* support) as well as *Keras* were used for model creation and training. This section describes the evaluation and training procedures, as well as the techniques used to improve model performance.

A combination of hold-out and modified cross-validation was used to evaluate the models. Data from four drivers were randomly selected for training, while data from the remaining two drivers were reserved for validation, resulting in a 68%/32% split between

training and validation. To ensure model generalization to new drivers, leave-one-driver-out cross-validation was performed, where each model was trained six times using data from five drivers for training and the remaining driver's data for testing. This approach provided an accurate estimate of the generalization performance of each model.

For the vision-based *CNN* model, due to memory constraints, the *TensorFlow Data API* was used to generate *TensorFlow* datasets for training and validation. Images were rescaled to a range of [0, 1] and a batch size of 16 was used. The *Adam Optimizer* with a learning rate of 10^{-4} and the *SparseCategoricalCrossEntropy* loss function were chosen. A early stopping mechanism was implemented to prevent overfitting by stopping training if the validation loss did not improve for three consecutive epochs.

Input sequences of absolute image trajectories and sensor data were used for the recurrent networks. Initially, a sequence length of two seconds with a one-second overlap was chosen, but this configuration resulted in poor generalization. A new configuration of sequence length of 3 seconds and a 0.5-second step was adopted, which significantly improved performance. A custom data generator was developed using the *TensorFlow Sequence API* to load and fill image sequences and handle sensor data. The training configuration, including optimizer, learning rate, and loss function, remained the same as the *CNN*, but the batch size was adjusted to 8 sequences.

During initial training, overfitting due to unbalanced class distribution was observed. The models tended to predict the dominant class, *normal_driving*, resulting in a validation accuracy that reflected the class distribution. To address this, the loss function was weighted using the inverse class distribution, forcing the model to pay more attention to the minority classes and achieving a more balanced learning process.

Another observation was that the models, especially the *CNN*, performed best in the first epoch, but declined in subsequent epochs, likely due to high redundancy in the data. To mitigate this, temporal downsampling was applied, reducing the sampling rate from 30 frames per second to 5 frames per second. This adjustment allowed for training over multiple epochs, allowing for better monitoring of validation loss trends.

3. Results

Several models were trained to evaluate the potential improvement of a vision-based driver distraction detection method by incorporating a recurrent neural network and sensor data. The models used in this study were the vision-based model (referred to as *CNN*), the vision-based recurrent model (referred to as *CNN-LSTM*), and the multimodal recurrent model that processed sequences of images and sensor data (referred to as *CNN-LSTM-S*). To allow a fair comparison with the recurrent networks, an additional approach was used to obtain the *CNN* predictions for image sequences without training a new model. This involved averaging the class probabilities from the *CNN* for each image within a sequence, and the resulting approach was named *CNN Sequential*. Consequently, the validation of the *CNN* was performed on individual images, while the validations for the remaining three approaches were performed on image (and sensor) sequences. Since the initial weights for each model training were chosen randomly, identical results were never obtained. To obtain statistically significant results, each model was trained five times in a row, and the results of each simulation run were averaged. The standard deviation of the corresponding scores was also reported for each result. A high variance in the scores could indicate that the models are stuck in local maxima due to the stochastic nature of the training, or are experiencing overfitting.

The results of the different models on the validation set are presented in Table 2. For each model, the metrics of overall accuracy, precision (macro average), recall (macro average), F1 score (macro average), balanced accuracy, and ROC area under the curve (AUC) were computed. The *CNN* showed an overall accuracy of 89% and a precision of 85%. However, the recall was slightly lower at 71%, indicating that the model had some difficulty in correctly assigning certain classes. This observation is also reflected in the F1 score and the balanced accuracy, which is insensitive to class imbalance, unlike the

ordinary accuracy. The high ROC AUC value of 0.94 suggests that the model had a strong discriminative power across classes. Looking at the table, it can be seen that *CNN* was the weakest in all metrics. In terms of accuracy and precision, the sequential approach with the CNN (*CNN Sequential*) achieved the highest scores of 92% each. The *CNN-LSTM-S* model achieved the best recall with 77%, but it showed significant variation across training runs, as indicated by a standard deviation of 5%. For the F1 score and balanced accuracy, the *CNN-LSTM-S* model also showed the highest average results, but with a relatively high standard deviation. Nevertheless, these results suggest that distractions can be detected more accurately when the temporal component is taken into account. However, determining whether the *CNN-LSTM-S* model reliably outperforms the *CNN-LSTM* model is challenging due to the considerable variability in scores across multiple training runs.

Table 2. Results of different models on the validation set for multi-class classification. Each cell represents the mean and standard deviations of the scores obtained from 5 training runs.

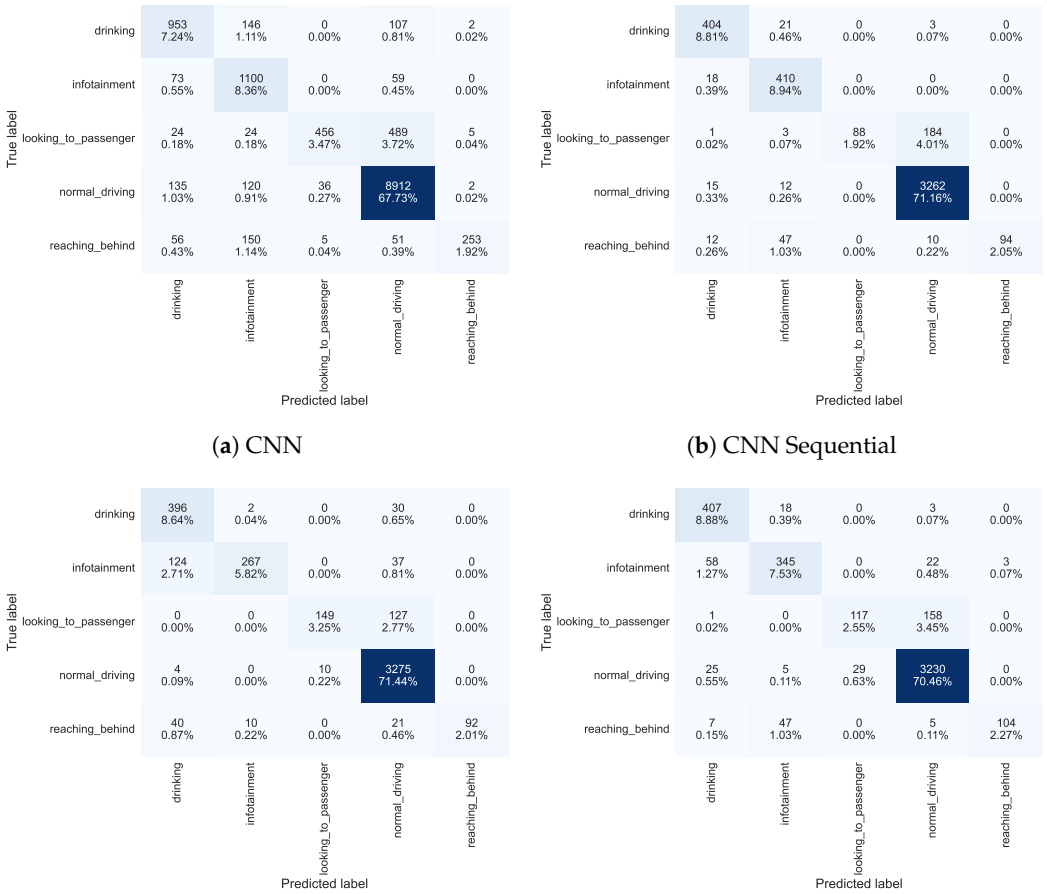
	CNN	**CNN Sequential**	**CNN-LSTM**	**CNN-LSTM-S**
Accuracy	0.89 (0.01)	0.92 (0.01)	0.91 (0.02)	0.91 (0.02)
Precision	0.85 (0.03)	0.92 (0.02)	0.88 (0.03)	0.86 (0.02)
Recall	0.71 (0.03)	0.75 (0.03)	0.73 (0.03)	0.77 (0.05)
F1 Score	0.75 (0.03)	0.78 (0.04)	0.77 (0.05)	0.79 (0.04)
Balanced Accuracy	0.71 (0.03)	0.75 (0.03)	0.73 (0.03)	0.77 (0.05)
ROC AUC	0.94 (0.01)	0.99 (0.00)	0.97 (0.00)	0.97 (0.01)

In Table 3, the results of the individual models on the validation set are presented, this time divided by the individual classes. Notably, high standard deviations are often observed for all metrics for all classes except *normal_driving*. These variations suggest that the final trained models behave very differently and may have ended up in different local minima due to initial random weights. However, the table shows that all models show the best classification performance for the *normal_driving* class. Notably, this class has a particularly high recall, indicating a high number of true positives and minimal misclassification of other classes as *normal_driving*. The *CNN Sequential* model even achieves a recall of 100%, perfectly classifying all *normal_driving* sequences. The *CNN Sequential* model also performs impressively well for the *drinking*, *infotainment*, and *reaching_behind* classes. However, for the class *looking_to_passenger*, the sequential approach with the *CNN* model encounters difficulties. The recall and F1 scores show that averaging the probabilities of the *CNN* classes leads to worse results. This degradation could be due to sequences in which the driver frequently swivels his head back and forth to briefly focus on the road while reading the sentences. In such cases, the model tends to classify some frames with high confidence as *normal_driving*, resulting in an incorrect average classification of *normal_driving* for the whole sequence. For the *looking_to_passenger* class, the recurrent networks (*CNN-LSTM* and *CNN-LSTM-S*) achieve the best results. However, even here, the average recall rates are relatively low, 58% and 62%, which may be due to the same problem. When comparing the performance of *CNN-LSTM* and *CNN-LSTM-S* across classes, several similarities can be observed. However, there is a notable difference in the *reaching_behind* class. The *CNN-LSTM* model achieves a precision of 100%, which means that all sequences classified as *reaching_behind* really belong to this class. On the other hand, when the additional sensor data are considered, the average precision drops to 91% (with a standard deviation of 8%), which is surprising.

Table 3. Results of different models on the validation set for different classes. Each cell represents the mean and standard deviations of the scores obtained from 5 training runs.

	CNN	CNN Sequential	CNN-LSTM	CNN-LSTM-S
		class: drinking		
Precision	0.82 (0.13)	0.91 (0.12)	0.71 (0.12)	0.76 (0.12)
Recall	0.79 (0.03)	0.95 (0.02)	0.94 (0.05)	0.92 (0.05)
F1 Score	0.80 (0.05)	0.93 (0.06)	0.80 (0.08)	0.83 (0.07)
		class: infotainment		
Precision	0.79 (0.04)	0.89 (0.04)	0.83 (0.09)	0.82 (0.08)
Recall	0.76 (0.14)	0.85 (0.14)	0.57 (0.24)	0.67 (0.22)
F1 Score	0.77 (0.07)	0.86 (0.07)	0.65 (0.23)	0.71 (0.14)
		class: looking_to_passenger		
Precision	0.91 (0.02)	0.99 (0.02)	0.93 (0.03)	0.85 (0.06)
Recall	0.40 (0.06)	0.22 (0.08)	0.58 (0.10)	0.62 (0.21)
F1 Score	0.56 (0.06)	0.36 (0.12)	0.71 (0.06)	0.70 (0.14)
		class: normal_driving		
Precision	0.91 (0.01)	0.93 (0.01)	0.95 (0.01)	0.96 (0.02)
Recall	0.98 (0.01)	1.00 (0.00)	0.99 (0.00)	0.98 (0.01)
F1 Score	0.95 (0.00)	0.96 (0.00)	0.97 (0.00)	0.97 (0.01)
		class: reaching_behind		
Precision	0.80 (0.11)	0.89 (0.09)	1.00 (0.00)	0.91 (0.08)
Recall	0.63 (0.09)	0.74 (0.11)	0.56 (0.05)	0.63 (0.13)
F1 Score	0.70 (0.04)	0.80 (0.05)	0.72 (0.04)	0.74 (0.09)

The visualization of the confusion matrices for the different models is shown in Figure 7. It is important to note that each confusion matrix represents the results of a single model, as it is difficult to capture the performance across multiple training runs. Therefore, for each model, the confusion matrix is generated using the model from the training run with the median F1 score. Additionally, it should be emphasized that the absolute numbers in the matrices are only comparable within Figure 7b–d as a different validation set was used in Figure 7a. Despite the significant differences in model behavior indicated by the metrics, similar patterns can be seen in the confusion matrices. In each matrix, the majority of samples are correctly classified, resulting in a slightly denser diagonal line from top left to the bottom right. However, due to the class imbalance, there is an outlier in the *normal_driving* class, which contains by far the largest number of samples. Across all models, the most frequent misclassifications occur in the *looking_to_passenger* class, with many samples classified as *normal_driving*. As mentioned before, this is due to the constant head-turning of the drivers while reading the sentences. In addition, especially in the *CNN-LSTM* model, there is a lot of confusion between the *infotainment* and *drinking* classes. Differentiating between samples where the driver is reaching for the infotainment system and when they are reaching for a water bottle or putting it down seems to be a challenge for the models. Both the *CNN Sequential* and *CNN-LSTM-S* models show 47 misclassifications, where *reaching_behind* sequences are classified as *infotainment*. These two classes involve manual distractions where the driver moves his right hand, which likely contributed to the difficulty in distinguishing between them for these specific models.



(**a**) CNN (**b**) CNN Sequential

(**c**) CNN-LSTM (**d**) CNN-LSTM-S

Figure 7. Confusion matrices for the individual models. For each model type, the training run that achieved the median F1 score was selected.

All models are designed for multi-class classification, where the output corresponds to either a distraction class or the *normal_driving* class. Consequently, in the previous evaluation metrics, errors between two distraction classes are weighted the same as errors between a distraction class and *normal_driving*. To evaluate the performance of the models when distinguishing only between distracted and normal driving, the problem was transformed into a binary classification task. However, no new models with modified architectures were trained specifically for this task. Instead, the results from multi-class classification were adapted to binary classification, which essentially aggregates all distracted classes. It is important to note that this approach is different from training new models explicitly designed to discriminate between these two classes. The overall results of the binary classification are in Table 4. Since confusion between different distraction classes is no longer a factor in binary classification, the results show some improvement over Table 2. The observed accuracies range from 92% to 96%, and the precision values are between 93% and 96%. Once again, the *CNN* model had the lowest scores for all metrics, while the time-dependent approaches performed better. In addition, it can be observed that the LSTM-based models have slightly higher recall and balanced accuracy, both at 93%, compared to the sequential CNN approach. However, there is no significant difference between *CNN-LSTM* and *CNN-LSTM-S*.

Table 4. Results of different models on the validation set for binary classification. Each cell represents the mean and standard deviations of the scores obtained from 5 training runs.

	CNN	CNN Sequential	CNN-LSTM	CNN-LSTM-S
Accuracy	0.92 (0.01)	0.95 (0.01)	0.96 (0.00)	0.95 (0.01)
Precision	0.93 (0.01)	0.96 (0.00)	0.96 (0.00)	0.95 (0.01)
Recall	0.88 (0.01)	0.91 (0.01)	0.93 (0.01)	0.93 (0.02)
F1 Score	0.90 (0.01)	0.93 (0.01)	0.94 (0.01)	0.94 (0.01)
Balanced Accuracy	0.88 (0.01)	0.91 (0.01)	0.93 (0.01)	0.93 (0.02)
ROC AUC	0.94 (0.00)	0.99 (0.00)	0.99 (0.01)	0.99 (0.00)

In addition to the hold-out validation, a leave-one-driver-out cross-validation was performed. The purpose of this cross-validation was to assess the sensitivity of the models to validation data from different drivers. The results of this method are visualized in Figure 8 using boxplots, which display various metrics for different models. Each boxplot represents the distribution of a particular score across different validation drivers. Notably, outliers can be observed in all metrics (except precision) for all models except *CNN-LSTM*. These outliers are consistently are associated with the same validation driver, indicating a failure of the models to generalize. Further examination of the dataset revealed that the camera images for this driver had darker brightness values compared to others, likely due to unfavorable weather-related lighting conditions during recording. In addition, this particular driver was the only one wearing a black long-sleeved top, which, combined with the poor lighting, made it difficult to recognize the arms. These factors suggest that the models struggled with this particular driver, as the training data from the five drivers with better lighting conditions differed significantly from the validation data with the driver with poor lighting conditions. However, looking at the boxplots of the *CNN-LSTM* model (shown in green), it is clear that this model was still able to handle the validation data from this difficult driver, as there are no outliers (although the performance is not at the same level as the other drivers). This observation suggests that the LSTM-based model has a higher tolerance for darker images due to its consideration of temporal dependencies, allowing for more accurate class predictions by processing multiple images. Surprisingly, the *CNN-LSTM-S* model, which incorporates sensor data in addition to visual data, encountered similar difficulties as the *CNN* and *CNN Sequential* models. In this case, the additional sensor data seemed to distract the model rather than help it.

A trend is observable in the boxplots, indicating increasing performance across the models, with the *CNN* showing the poorest performance in each metric, followed by the *CNN Sequential* model, and then the *CNN-LSTM* model. The *CNN-LSTM-S* model indicates the highest 75% quantiles and maximum values for each score. This trend is also observed in the median values, except for precision, where the *CNN-LSTM* slightly outperforms the *CNN-LSTM-S* model. Nevertheless, disregarding the outliers, the boxplots suggest that the *CNN-LSTM* model, when considering sensor data, delivers the best performance in the leave-one-driver-out cross-validation.

Table 5 shows a tabular representation of the leave-one-driver-out cross-validation results. Each block of five rows corresponds to a train–test split where one driver's data are used for validation. The first column shows the results from the *CNN* model, which serves as the baseline for comparison. Subsequent columns show results from other models, with the absolute difference from the *CNN* score next to each score. The table clearly shows that the models using validation data from *driver_0* are responsible for the outliers in the boxplots. Only the *CNN-LSTM* model achieved reasonable results, with an accuracy of 83% and a balanced accuracy of 71%. Excluding the results of *driver_0*, the vision-based approach with the *CNN* consistently performed the worst. Averaging the *CNN* class probabilities over sequences (*CNN Sequential*) shows an improvement, with balanced accuracies showing improvements of 5–10%. However, the results of the two LSTM-based models suggest that the use of a time-dependent model can achieve better classification performance. Except for

a few cases, such as precision for *driver_2* and F1 score, precision and recall for *driver_5*, the *CNN-LSTM-S* model outperformed the others for all metrics. Nevertheless, no significant differences were observed with respect to the *CNN-LSTM* model without sensor data. When comparing the models without considering *driver_0*, the average F1 scores across all drivers provide the following insights: The *CNN* model achieved an average F1 score of 71.3%. By averaging the class probabilities alone, the sequential approach with CNN improved to an average F1 score of 77.9%. However, by incorporating the *CNN-LSTM* network, this value increased significantly to 87.0%. This indicates that the use of a recurrent network allows for more accurate detection of distractions compared to the simple sequential CNN approach. The final model, the recurrent multimodal *CNN-LSTM-S*, had the highest average F1 score of 90.1%, indicating an improved classification performance. It is important to note that F1 scores varied considerably among individual drivers, highlighting the need for a larger driver dataset to obtain statistically more reliable values.

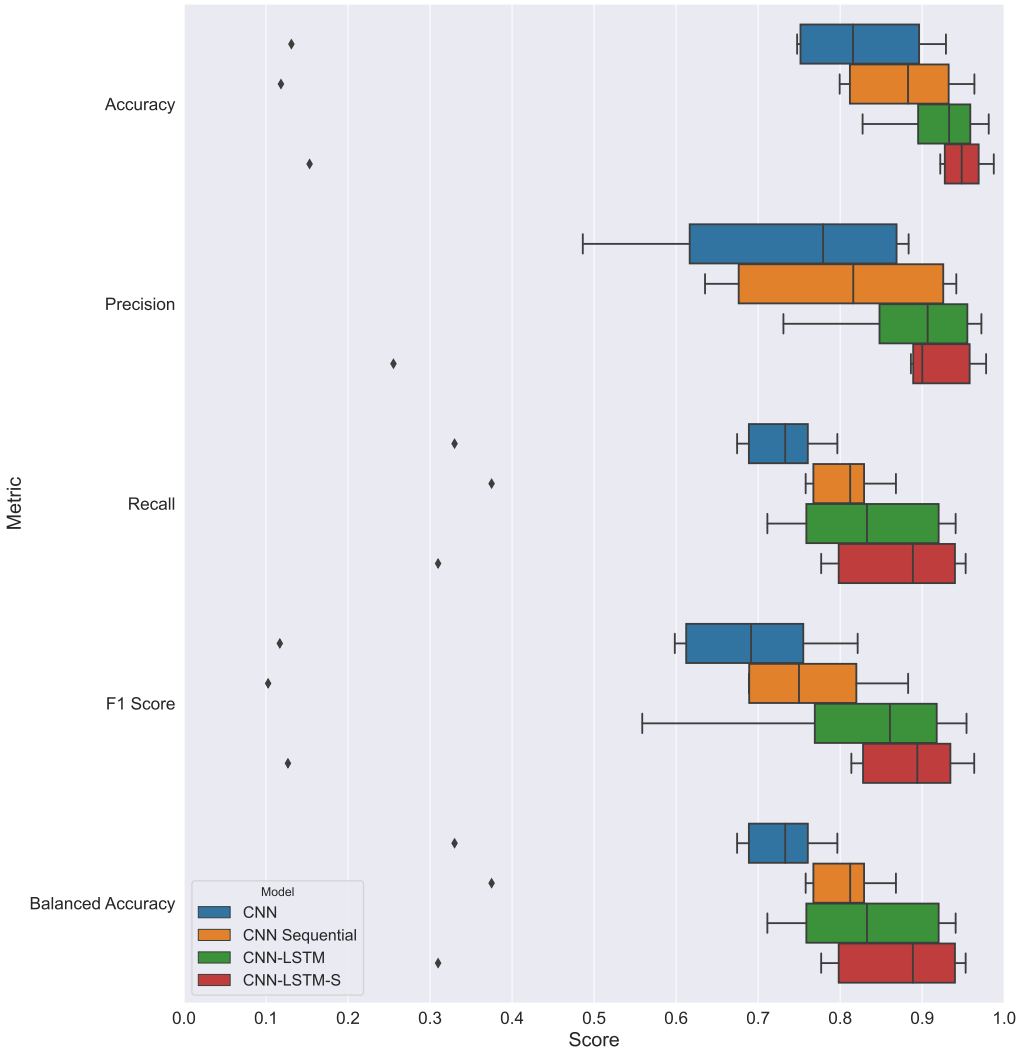


Figure 8. Results of the leave-one-driver-out cross-validation. A boxplot represents the scores across the different validation drivers.

Table 5. Leave-one-driver-out cross-validation results of different models for different validation drivers. Each cell represents the score and the score difference to the baseline CNN model.

	CNN	CNN Sequential	CNN-LSTM	CNN-LSTM-S
validation driver: driver_0				
Accuracy	0.13	0.12 (−0.01)	0.83 (+0.70)	0.15 (+0.02)
Precision	0.49	0.64 (+0.15)	0.73 (+0.24)	0.26 (−0.23)
Recall	0.33	0.38 (+0.05)	0.71 (+0.38)	0.31 (−0.02)
F1 Score	0.12	0.10 (−0.02)	0.56 (+0.44)	0.13 (+0.01)
Balanced Accuracy	0.33	0.38 (+0.05)	0.71 (+0.38)	0.31 (−0.02)
validation driver: driver_1				
Accuracy	0.91	0.94 (+0.03)	0.98 (+0.07)	0.99 (+0.08)
Precision	0.88	0.90 (+0.02)	0.97 (+0.09)	0.98 (+0.10)
Recall	0.73	0.80 (+0.07)	0.94 (+0.21)	0.95 (+0.22)
F1 Score	0.76	0.81 (+0.05)	0.95 (+0.19)	0.96 (+0.20)
Balanced Accuracy	0.73	0.80 (+0.07)	0.94 (+0.21)	0.95 (+0.22)
validation driver: driver_2				
Accuracy	0.87	0.92 (+0.05)	0.89 (+0.02)	0.94 (+0.07)
Precision	0.84	0.94 (+0.10)	0.83 (−0.01)	0.89 (+0.05)
Recall	0.67	0.76 (+0.09)	0.76 (+0.09)	0.86 (+0.19)
F1 Score	0.73	0.82 (+0.09)	0.75 (+0.02)	0.87 (+0.14)
Balanced Accuracy	0.67	0.76 (+0.09)	0.76 (+0.09)	0.86 (+0.19)
validation driver: driver_3				
Accuracy	0.93	0.96 (+0.03)	0.96 (+0.03)	0.97 (+0.04)
Precision	0.88	0.93 (+0.05)	0.96 (+0.08)	0.98 (+0.10)
Recall	0.80	0.87 (+0.07)	0.90 (+0.10)	0.91 (+0.11)
F1 Score	0.82	0.88 (+0.06)	0.92 (+0.10)	0.94 (+0.12)
Balanced Accuracy	0.80	0.87 (+0.07)	0.90 (+0.10)	0.91 (+0.11)
validation driver: driver_4				
Accuracy	0.75	0.80 (+0.05)	0.94 (+0.19)	0.95 (+0.20)
Precision	0.72	0.73 (+0.01)	0.89 (+0.17)	0.90 (+0.18)
Recall	0.77	0.83 (+0.06)	0.93 (+0.16)	0.95 (+0.18)
F1 Score	0.65	0.69 (+0.04)	0.90 (+0.25)	0.92 (+0.27)
Balanced Accuracy	0.77	0.83 (+0.06)	0.93 (+0.16)	0.95 (+0.18)
validation driver: driver_5				
Accuracy	0.76	0.85 (+0.09)	0.92 (+0.16)	0.92 (+0.16)
Precision	0.58	0.66 (+0.08)	0.93 (+0.35)	0.90 (+0.32)
Recall	0.73	0.83 (+0.10)	0.77 (+0.04)	0.78 (+0.05)
F1 Score	0.60	0.69 (+0.09)	0.82 (+0.22)	0.81 (+0.21)
Balanced Accuracy	0.73	0.83 (+0.10)	0.77 (+0.04)	0.78 (+0.05)

4. Discussion

The validation results obtained from both the hold-out method and the leave-one-driver-out cross-validation provide valuable insights into the performance of the proposed models. The hold-out evaluations suggest a slight trend indicating that the inclusion of a temporal component improves the reliable detection of distractions. However, in certain cases, the sequential approach using a simple CNN outperformed more complex models in terms of classification accuracy. The evaluation results did not conclusively prove the superiority of recurrent neural networks, such as LSTM networks, for distraction detection. In particular, the hold-out method, which maintains a consistent split between training and validation, showed high variability in results across repeated training sessions. This was particularly evident when analyzing metrics for individual classes, where significant standard deviations were observed. This variability is likely due to the stochastic nature of neural networks, leading to different local minima during each training run.

Leave-one-driver-out cross-validation was used to evaluate the sensitivity of the models to the specific combinations of driver data across training and test sets. The results showed that different validation drivers occasionally produced different results. For example, all models except the *CNN-LSTM* model struggled with validation data from a driver experiencing difficult lighting conditions. However, this form of validation highlighted a trend where time-dependent LSTM networks outperformed the CNN models, both with and without a sequential approach. In addition, the leave-one-driver-out cross-validation suggested that the LSTM network incorporating sensor data tended to outperform the LSTM network processing only image sequences.

Despite these findings, it remains inconclusive whether the inclusion of sensor data significantly improves prediction performance. Further research is needed to draw definitive conclusions about the impact of sensor data on driver distraction detection. Such research should include a larger dataset with a more diverse set of drivers and a greater emphasis on feature engineering. In addition, a broader range of model architectures would need to be explored to reliably address this issue.

5. Conclusions

This study focused on the development of a recurrent multimodal approach for driver distraction detection and investigated whether classical vision-based methods using CNNs could be improved by incorporating LSTM networks and sensor data. A custom dataset was created with six drivers performing different distracting activities. A *CNN* model using a pre-trained *VGG16* served as the baseline. This study introduced a *CNN-LSTM* model to capture temporal dependencies and a multimodal *CNN-LSTM-S* model that combined image sequences with sensor data. Comparisons were made using a *CNN Sequential* approach, where *CNN* outputs for each frame were averaged across sequences.

The results showed that incorporating temporal dependencies improved classification performance. While the *CNN Sequential* approach slightly outperformed LSTM-based models in accuracy, high variability was observed across training runs. Leave-one-driver-out cross-validation showed that all models except the *CNN-LSTM* struggled with validation data from a driver with darker images, highlighting the importance of considering multiple images for reliable detection. The recurrent approaches, especially the LSTM-based models, showed significantly better performance in such scenarios.

The inclusion of an LSTM network improved the accuracy of the vision-based approach, and the sensor data also showed a trend toward improvement. However, further research is needed to validate this trend. The primary objective of this work was to compare different technologies rather than to develop a model for production use. Consistency in the recordings was maintained, but the results underscore the need for a larger, more diverse dataset for stable results. For practical use, more recordings should be made with different drivers, vehicles, routes, and recording conditions. A diverse dataset would allow the model to better generalize to new data, contributing to the development of a robust, practical model suitable for real-world implementation.

This study considered a limited set of distraction classes, justified by the practical need to capture sufficient samples within a short recording time. However, a real-world model would need to include a broader range of distractions. Despite the limited number of classes, the class imbalance was a challenge, with the *normal_driving* class dominating. Loss function weighting was used to address this, but further exploration of methods such as artificial upsampling is recommended. The research was also limited in terms of feature engineering, focusing on standard procedures such as scaling and normalizing the data. Future work could include more advanced feature extraction and exploration of alternative model architectures such as bidirectional LSTM networks or different CNN architectures such as *VGG19*, *InceptionV3*, *ResNet*, or *DenseNet*.

In summary, AI-based driver distraction detection systems hold potential for future implementation in vehicles. With further adaptation and improvement, these systems could accurately detect distractions and issue warnings, thereby reducing distraction-related accidents. Such systems are especially critical for partially automated vehicles, ensuring driver alertness and facilitating safe control takeovers, ultimately contributing to overall road safety.

Author Contributions: Conceptualization, M.C. and G.O.; Methodology, M.C.; Software, M.C.; Writing—original draft, M.C.; Writing—review & editing, M.C.; Supervision, G.O.; Project administration, G.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Data Availability Statement: The dataset presented in this study is available upon request from the corresponding author. The data are not publicly available due to privacy reasons.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

Some additional visualizations for the exploratory data analysis described in Section 2.3 are presented in this appendix.

In Figure A1, in addition to the boxplots in Figure 3, a histogram is presented to illustrate the distribution of distraction durations across different distraction tasks. This visualization reveals that both of the classes, *drinking* and *infotainment*, not only have longer durations but also exhibit the widest range of durations among all distractions.

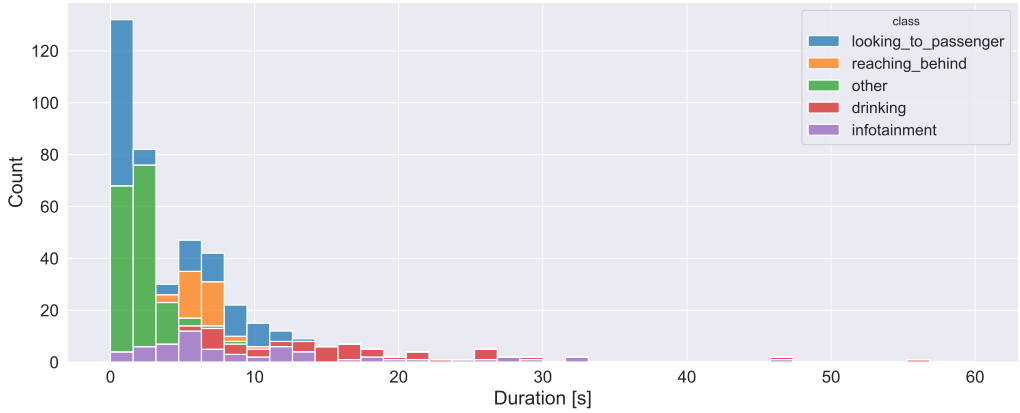
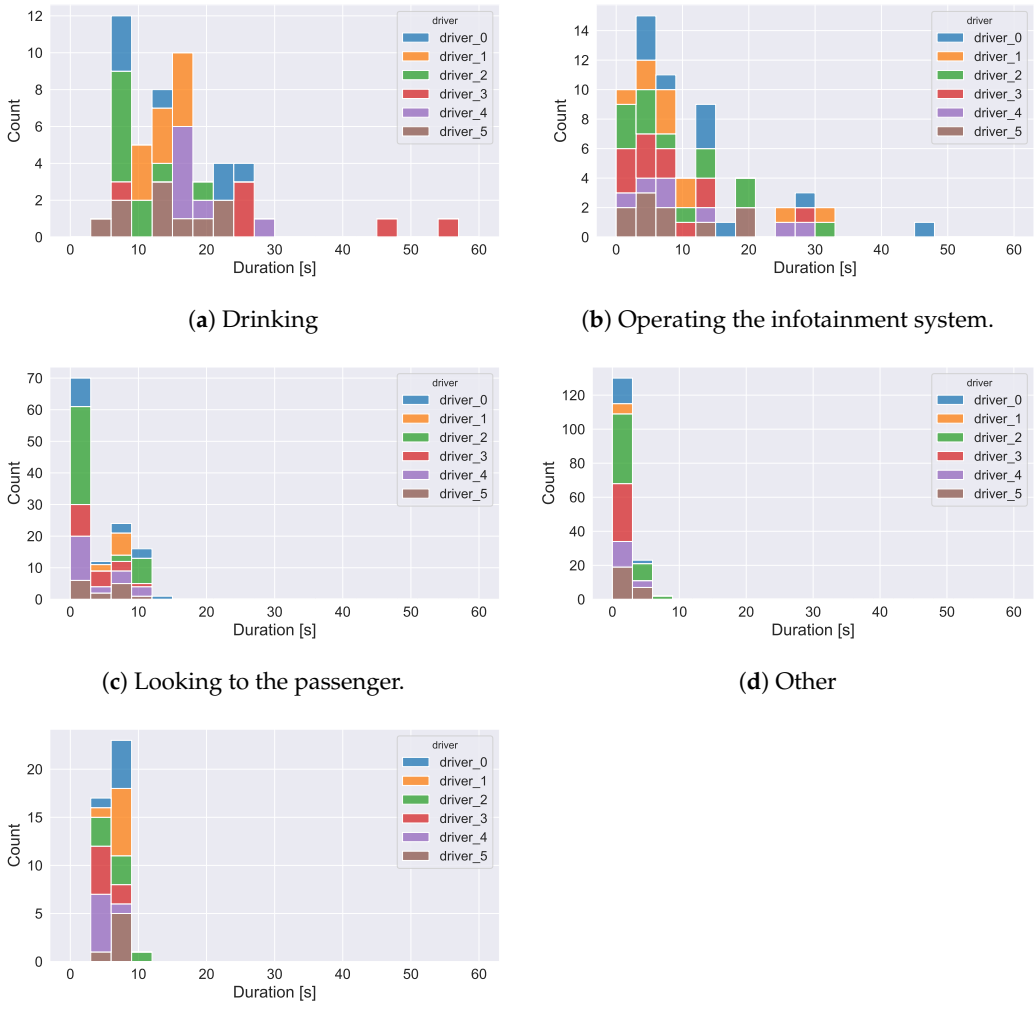


Figure A1. Histogram with durations of distracting activities.

In Figure A2, the durations of various types of distractions are depicted using multiple histograms. This visualization clearly illustrates the impact of different drivers' behaviors on distraction durations. The data distribution across the bins shows significant variations among the drivers. Specifically, when examining the drinking class in Figure A2a, it is evident that driver 3 (depicted in red) experiences longer periods of distraction compared to the other drivers. Moreover, driver 3 is responsible for the appearance of two outliers in the data.



(**a**) Drinking

(**b**) Operating the infotainment system.

(**c**) Looking to the passenger.

(**d**) Other

(**e**) Reaching behind

Figure A2. Visualization of histograms of durations of different distractions separated by driver.

References

1. Regan, M.A.; Lee, J.D.; Young, K. *Driver Distraction: Theory, Effects, and Mitigation*; CRC Press: Boca Raton, FL, USA, 2008.
2. European Road Safety Observatory. *Road Safety Thematic Report—Driver Distraction*; Technical Report; European Commission: Brussels, Belgium, 2022.
3. Kinnear, N.; Stevens, A. *The Battle for Attention: Driver Distraction—A Review of Recent Research and Knowledge*; Report; Institute of Advanced Motorists: Welwyn Garden City, UK, 2015.
4. World Health Organization. *Global Status Report on Road Safety 2018*; World Health Organization: Geneva, Switzerland, 2018; p. 403.
5. TRL; TNO; Rapp Trans. *Study on Good Practices for Reducing Road Safety Risks Caused by Road User Distractions*; Technical Report; European Commission: Brussels, Belgium, 2015.
6. Kashevnik, A.; Shchedrin, R.; Kaiser, C.; Stocker, A. Driver distraction detection methods: A literature review and framework. *IEEE Access* **2021**, *9*, 60063–60076. [CrossRef]
7. Omerustaoglu, F.; Sakar, C.O.; Kar, G. Distracted driver detection by combining in-vehicle and image data using deep learning. *Appl. Soft Comput.* **2020**, *96*, 106657. [CrossRef]
8. State Farm Distracted Driver Detection Competition. Online Resource. Available online: https://www.kaggle.com/c/state-farm-distracted-driver-detection (accessed on 16 September 2024).
9. Huang, G.; Liu, Z.; Van Der Maaten, L.; Weinberger, K.Q. Densely connected convolutional networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26 July 2017; pp. 4700–4708.
10. He, K.; Zhang, X.; Ren, S.; Sun, J. Deep residual learning for image recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016; pp. 770–778.
11. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. *arXiv* **2015**, arXiv:abs/1512.00567.

12. Mafeni Mase, J.; Chapman, P.; Figueredo, G.P.; Torres Torres, M. Benchmarking deep learning models for driver distraction detection. In Proceedings of the Machine Learning, Optimization, and Data Science: 6th International Conference, LOD 2020, Siena, Italy, 19–23 July 2020; Revised Selected Papers; Part II 6; Springer: Berlin/Heidelberg, Germany, 2020; pp. 103–117.
13. Baheti, B.; Talbar, S.; Gajre, S. Towards Computationally Efficient and Realtime Distracted Driver Detection With MobileVGG Network. *IEEE Trans. Intell. Veh.* **2020**, *5*, 565–574. [CrossRef]
14. Qin, B.; Qian, J.; Xin, Y.; Liu, B.; Dong, Y. Distracted Driver Detection Based on a CNN With Decreasing Filter Size. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 6922–6933. [CrossRef]
15. Wang, X.; Xu, R.; Zhang, S.; Zhuang, Y.; Wang, Y. Driver distraction detection based on vehicle dynamics using naturalistic driving data. *Transp. Res. Part C Emerg. Technol.* **2022**, *136*, 103561. [CrossRef]
16. Streiffer, C.; Raghavendra, R.; Benson, T.; Srivatsa, M. Darnet: A deep learning solution for distracted driving detection. In Proceedings of the 18th acm/ifip/usenix Middleware Conference: Industrial Track, Las Vegas, NV, USA, 11–15 December 2017; pp. 22–28.
17. Zheng, A.; Casari, A. *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2018.
18. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv* **2014**, arXiv:1409.1556.
19. Srinivasan, K.; Garg, L.; Datta, D.; Alaboudi, A.A.; Jhanjhi, N.; Agarwal, R.; Thomas, A.G. Performance comparison of deep cnn models for detecting driver's distraction. *CMC-Comput. Mater. Contin.* **2021**, *68*, 4109–4124. [CrossRef]
20. Landup, D. Don't Use Flatten()—Global Pooling for CNNs with TensorFlow and Keras. Online Resource. Available online: https://stackabuse.com/dont-use-flatten-global-pooling-for-cnns-with-tensorflow-and-keras/ (accessed on 16 September 2024).
21. Géron, A. *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow*; O'Reilly Media, Inc.: Sebastopol, CA, USA, 2022.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.