CNN-LSTM architecture for predicting hazardous driving situations

Using vehicle and weather data

Master's thesis in Computer science and engineering

NOOMI LINDBLAD

STEFANI PLATAKIDOU

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2023

Master's thesis 2023

CNN-LSTM architecture for predicting hazardous driving situations

Using vehicle and weather data

NOOMI LINDBLAD
STEFANI PLATAKIDOU

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2023

CNN-LSTM architecture for predicting hazardous driving situations
Using vehicle and weather data
NOOMI LINDBLAD
STEFANI PLATAKIDOU

Supervisor: Alexandru Gheorghiu, Department of Computer Science and Engineering
Advisor: Therese Gardell, Volvo GTT
Examiner: Devdatt Dubhashi, Department of Computer Science and Engineering

Master's Thesis 2023
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2023

CNN-LSTM architecture for predicting hazardous driving situations
Using vehicle and weather data
NOOMI LINDBLAD
STEFANI PLATAKIDOU
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

This study aims to investigate how a CNN-LSTM model can be used together with recorded vehicle data from trucks and external weather data in order to predict a hazardous driving situation. The dataset consists of three-second-long driving snippets from customer and development trucks registered within Europe. The combination of a CNN and LSTM was implemented using two different architectures, one parallel and one sequential. The models were compared to a Random Forest classifier, as well as to a CNN and an LSTM individually. All models were evaluated with the complete dataset, data without weather features, and noisy data. The results from the complete dataset revealed that the Random Forest classifier achieved the highest accuracy of 92%, followed by the parallel CNN-LSTM with an accuracy of 81%. All models except the Random Forest classifier performed better with noisy data. The outcome of the thesis challenges the initial hypothesis that a CNN-LSTM is the optimal model given the context.

Keywords: Data science, Machine learning, LSTM, CNN, Vehicle data, Hazardous driving situation, Deep learning.

Acknowledgements

We would like to thank Therese Gardell, Sofia Karlsson, and Rached Dardouri from Volvo Group for their support and insights. You have been of immense support throughout this thesis, from data collection to the finishing touches.

Finally, we want to express our gratitude to Alexandru Gheorghiu, our academic supervisor. Thank you for your assistance and guidance, it has been of great help for this thesis.

Noomi Lindblad and Stefani Platakidou, Gothenburg, 2023-05-30

Contents

List of Figures

List of Tables

List of Acronyms

HMI Human Machine Interface

ADAS Advanced Driver Assistance Systems

TSC Time Series Classification

DL Deep Learning

CNN Convolutional Neural Network

LSTM Long Short-Term Memory

RF Random Forest

MLP-NN Multilayer Perceptron Neural Network

RNN Recurrent Neural Network

CEC Constant Error Carousel

TP True Positive

FN False Negative

TN True Negative

FP False Positive

TPR True Positive Rate

FPR False Positive Rate

FNR False Negative Rate

ROC Receiver Operating Characteristics

AUC Area Under the ROC Curve

CAN Controller Area Network

1
Introduction

Road traffic accidents are today one of the leading causes of death, and according to the Centers for Disease Control and Prevention, around 1.35 million people are killed every year around the world in road-related accidents (*Centers for Disease Control and Prevention* 2020). Road safety can be increased by using Advanced Driver Assistance Systems (ADAS) that predict hazardous traffic situations (Tigadi et al. 2016). ADAS include both active and passive safety systems, serving as aids to the driver in the course of driving. Together with the Human Machine Interface (HMI), ADAS not only react to vehicle sensors but also proactively communicate with the driver, thereby enhancing overall road safety.

During driving, the vehicle sensors of the Volvo trucks record sequences of data a few seconds before a function intervention. A function intervention is automatic braking that gets activated if a crash is predicted to occur. Information that is recorded includes the movement and speed of the truck. The hypothesis is that a machine learning model can utilize this information, together with weather data, to classify a situation as hazardous or not. Exploring what parameters are active prior to a function intervention and searching for trends and patterns is a way to identify a hazardous driving situation. In this context, a hazardous situation is represented either by automatic braking of the vehicle and no crash, or automatic braking and a crash. Implementing the final model so that it provides a message to the HMI, could help the driver make safer choices.

1.1 Project Aim

The objective of the thesis is to identify hazardous and non-hazardous driving situations, in the hopes of implementing this to increase road safety. Additionally, informing the driver of such a situation can help decrease road accidents. The problem at hand is modeled as a Time Series Classification (TSC) task, where the aim is to classify a given driving situation as either hazardous or non-hazardous. The available data is labeled and consists of time series data representing the values of vehicle sensors, for example speed and acceleration, at different moments of time. By investigating if there are any patterns or trends in the vehicle data together with external weather data a hazardous situation might be identified. Using a machine learning algorithm, the model can detect the necessary patterns to identify these situations.

The overall aim of the thesis is to explore if a combination of two machine learning classification models can be used to predict hazardous driving situations. The machine learning model will utilize recorded vehicle data together with external data about the weather. More precisely, this research aims to investigate whether a neural network architecture using a CNN combined with a LSTM network can improve the prediction of hazardous situations or not. Both a parallel and sequential version of a CNN-LSTM will be explored. To justify the use of a deep learning model, a comparison to a less complex machine learning model will be done. The less complex model of choice is a Random Forest (RF) classifier since it is suitable for time series data. Three baseline models will be compared to the hybrid CNN-LSTM models. The three baseline models are an RF classifier, a CNN, and an LSTM.

Several previous research papers about accident prediction and classification using vehicle data and weather data separately and together have been conducted, further discussed in Chapter 2. Similarly, research exists about how the combined CNN-LSTM model performs better than the individual models. Prior studies on accident prediction have not incorporated the CNN-LSTM model using vehicle and weather datasets for the specific purpose of accident prediction, whereas the thesis attempts to address this research gap.

The final goal of the thesis is to investigate how machine learning models can predict hazardous situations using vehicle and external weather data. In order to evaluate the performance of the models two additional experiments will be conducted. First, the models will be trained and tested without weather data in order to investigate whether the additional data improves the models or not. Second, the models will be trained and tested with noisy data to evaluate the models' robustness and generalization. The following questions can contribute to the goal of this research:

(a) *How does a hybrid CNN-LSTM approach compare to the two separate models in this context?*

(b) *How well does the classifier perform with weather data versus without weather data?*

(c) *Can the classifier improve its performance when noisy sensor data is introduced?*

1.2 Ethical Considerations

The main ethical consideration to take into account in this project is the privacy and consent of the driver. All data used from customer trucks were collected with consent from the customer. Further, every truck has a unique ID, if this is combined with the GPS information, a driver can potentially be identified together with their location. To avoid this, the ID and GPS location will be removed after merging the data to make it anonymous. Another ethical concern is whether the model can be biased. This can happen as a result of the constraints on the training data, as will be mentioned in Section 1.3. However, subject to these constraints, the model does not incorporate any personal information about the drivers. During the project, all data will be handled in a secure way that is protected from unauthorized access.

After using the collected data for this thesis' research and purpose, the data will be deleted to bypass unintended consequences.

1.3 Limitations

In this project, there are three primary constraints, with the initial one being the geographical origin of the gathered vehicle data. The provided dataset is limited to the European region, with the underlying assumption that these trucks are only driven within Europe. Other regions and countries might have different traffic conditions and regulations which in turn complicates the classification task. This constraint will affect the generalizability of the results and proposed model considering different countries have different cultures around traffic behavior.

The second limitation stems from the difference in how the data about hazardous and non-hazardous situations is collected. The data labeled as hazardous situations is collected from customer trucks with a great variation of vehicles and drivers. On the other hand, the data of the non-hazardous situations is collected from development trucks with less variation in terms of vehicles and drivers. This implies that the data from the non-hazardous situations differs from real-life driving since some of the active safety functions are turned off. The features representing the functions that are turned off will simply be removed from the dataset. This limitation might affect the generalization of the model considering the dataset might not fully represent the range of real-world driving scenarios and drivers. In addition, removing features may lead to a loss of information that might be relevant to the classification of the model.

The last limitation is the fact that the recorded vehicle data is only three seconds long, more described in Section 3.1.1. Thus, the number of data points for each driving instance is limited and might have some effect on the classification model.

2
Theory

The following sections introduce previous research within the field of accident prediction and TSC. Previous studies about crash prediction and their challenges are presented in Section 2.1, and previous studies about TSC and the three baseline models are presented in Section 2.2. Further, two different CNN-LSTM architectures are displayed in Section 2.3. Lastly, a brief discussion about the evaluation metrics is presented in Section 2.4.

2.1 Crash Prediction

Crash prediction is a well studied problem that has received significant attention in recent years. One of the first papers that attempted to predict crashes in traffic proposed using linear regression and Poisson regression in order to predict vehicle crashes on highways based on information about the number of vehicles from the Indian Toll Road and weather data (Jovanis & Chang 1986). The authors came to the conclusion that accidents increase with the number of vehicles on the road and snowfall. Further, Chen et al. (2012) studied the risk factors that increase intersection accidents. The authors used logistic regression and looked at data from previous intersection accidents in Australia between the years 2000 and 2009. They concluded that seven factors increase the intensity of the crash including the driver's age and gender.

More recently, Fu et al. (2022) hypothesized that data about the behavior of the driver can be used to predict the risk of a traffic accident. The study revealed that it is possible to predict traffic risks using an LSTM architecture. Although human error is often the main reason for accidents occurring, there can also be external factors that contribute to the risk of a specific situation. Yuan et al. (2018) created a Convolutional LSTM model to predict traffic accidents based on data from historical traffic crashes, weather, and road networks. Similarly, Kashifi et al. (2022) used different sources of data, such as weather, traffic information, and information about holidays to implement a hybrid of CNN and LSTM. The authors compared the hybrid CNN-LSTM to a logistic regression model and various deep neural networks like CNN and LSTM. The CNN-LSTM outperformed the baseline models and achieved an accuracy of 72 % and False Positive Rate (FPR) of 28%. In another study, Li et al. (2020) built a CNN-LSTM model for real-time crash risk prediction using traffic data, signal timing data, and weather data. The authors compared the

CNN-LSTM to five baseline models including an LSTM, XGBoost, and Bayesian Logistic Regression.

Furthermore, Osman & Hajij (2021) investigated how vehicle data can be used to predict crashes. The main objective of the study was to predict crashes using pattern recognition from observed driving behavior data. The authors compared three different DL algorithms, namely a Multilayer Perceptron Neural Network (MLP-NN), a CNN, and an LSTM to evaluate the predictions. Osman & Hajij (2021) hypothesis was that the driving pattern changes during and before a crash, measured by the motion of the vehicle. The dataset used included vehicle kinematics where every vehicle in the dataset either crashed, had a near-crash event, or had no crash or near-crash event. Vehicle kinematics refers to the motion and behavior of the vehicle without considering the forces that cause the motion. The study showed that both the LSTM and CNN models had 100% accuracy, whereas the MLP-NN had an accuracy of 96%.

2.2 Time Series Classification

Previous research showed that LSTM and CNN were successful at TSC tasks (Osman & Hajij 2021, Fullah Kamara et al. 2020). Fullah Kamara et al. (2020) used a combination of Contextual CNN and Contextual LSTM to classify both univariate and multivariate time series data. In this paper, contextual referred to long-term feature dependencies. The combined model was called CNTC and was tested on 44 different time series datasets provided by the University of California Riverside for benchmarking. Some examples of data used in this study were medical images, word synonyms, and symbols. The presented CNTC model indicated an improved result compared to the benchmark models.

Zhao et al. (2017) showed how a CNN can be used for time series classification using multivariate data. They claimed that CNNs can be a better approach when extracting features, compared to human-designed features. There has also been research combining LSTM with CNN to achieve higher accuracy for multivariate TSC. Karim et al. (2019) compared the combined model with other state-of-the-art models and an RF and two Support Vector Machines with different kernels. The authors used 35 different multivariate time series datasets. However, to the best of our knowledge, the CNN-LSTM model has never been applied to vehicle and weather data in order to predict hazardous driving situations.

2.2.1 Random Forest

As mentioned, one of the models that will be explored is the machine learning model called RF, which was first presented in a study by Breiman (2001). RF is an ensemble algorithm that contains multiple decision tree classifiers trained on subsets of the dataset. The algorithm uses bagging to select a random subset of the features for each decision tree to help increase the diversity of the classifier. The predicted results of each decision tree are averaged in order to achieve a higher accuracy of the

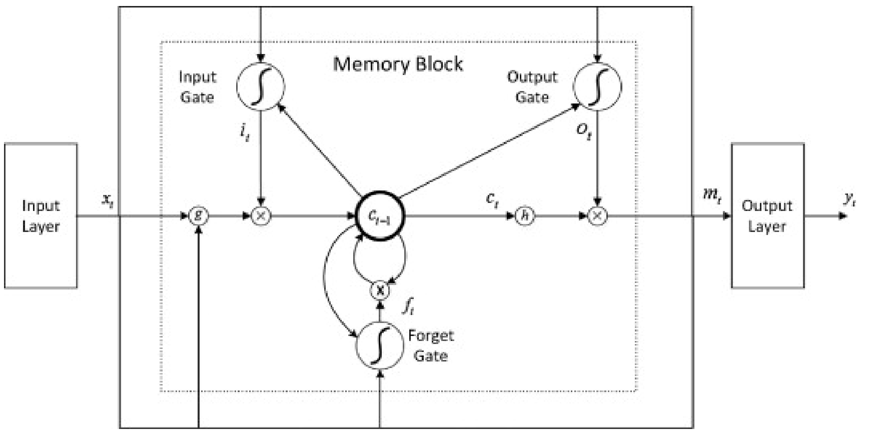


Figure 2.1: Sketch of the architecture of an LSTM-NN with a memory block containing one memory cell. Source: (Ma et al. 2015).

model. RF is known to be effective, robust to overfitting data, and achieving high performance. Further, RF is simple to implement and understand (Breiman 2001).

2.2.2 Long Short-Term Memory

The purpose of the LSTM is to predict sequential data in a more efficient way compared to the traditional Recurrent Neural Network (RNN). LSTM is based on the idea of the recurrent network, the difference is that LSTM uses a short-term memory based on the previous value in the time series together with a long-term memory that is represented by a weight based on all values in the time series (Hochreiter & Schmidhuber 1997). One issue with using RNNs is that during back propagation, the gradient is based not only on the previous layer but also on all the previous neurons. As a result, the recurring weight tends to increase beyond one during deep back propagation. This is problematic because a large recurring weight causes the gradient to become very small, which means it has little impact on the weights. This is known as the *vanishing gradient problem* (Bengio et al. 1994).

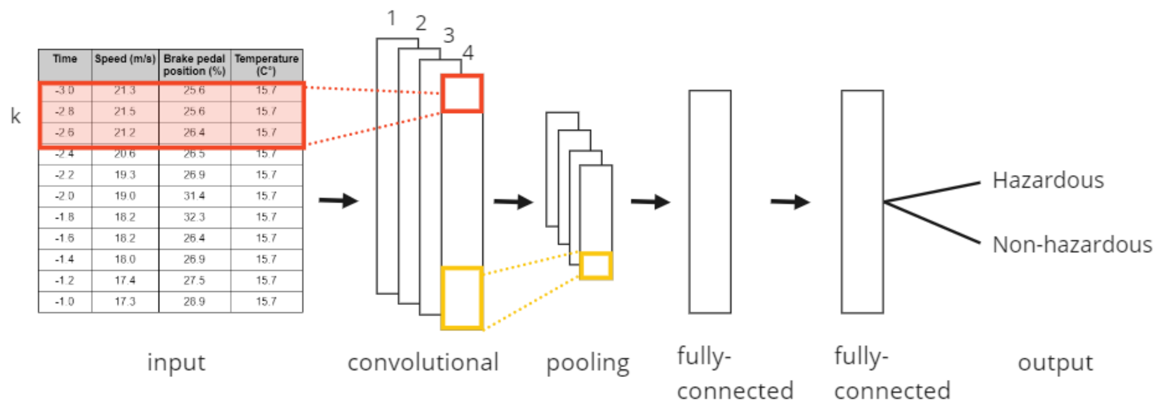
An LSTM Neural Network consists of three layers where the only hidden layer is a recurrent layer consisting of memory blocks, illustrated in Figure 2.1. The traditional memory blocks consist of memory cells, which is the unit where the computations take place. The memory cell has a core unit called Constant Error Carousel (CEC), and the activation of the CEC is the state of the cell (Hochreiter & Schmidhuber 1997, Ma et al. 2015, Gers 2001). The CEC will receive the long- and short-term memory from the previous cell and the input from the time series. Based on these, together with the weights and biases, the output will consist of the short- and long-term memory (Hochreiter & Schmidhuber 1997). As a result of the CEC that controls a constant error flow, the gradient will be less affected compared to a traditional RNN. Therefore, the LSTM is less susceptible to the vanishing gradient problem (Gers 2001).

The memory cell contains three gates: a forget, an input, and an output gate. Firstly, the *forget gate* generates a value that represents the recurrent connection in the

CEC state. This value is based on the output of the previous cell, together with the current input. These values will go through a sigmoid layer and the generated value between zero and one will be used in the CEC state as recurring weight. The weight from the forget gate is controlling the gradient. The purpose of the *input gate* is to determine what information that will be input to the CEC. The input will pass through a sigmoid function, accompanied by the activation function of the input gate. The value from the input gate will be one of the variables used in the CEC state. Subsequently, the previous state of the CEC is multiplied by the forget weight and added to the values from the input gate. When the cell state is updated the value will be filtered through a sigmoid function, followed by the activation function of the *output gate* which determines the final output of the memory cell (Abidogun 2005).

2.2.3 1D Convolutional Neural Networks

A CNN is a combination of three different ideas: convolution, pooling, and spatial sub-sampling (LeCun et al. 1995). Convolution and pooling are used to generate features of the raw data. The basic architecture of a 1D CNN consists of convolutional layers, pooling layers, and fully-connected layers, that are stacked upon each other. The convolutional layer consists of filters, also called kernels, that produce activation maps that extract features from the input. The filter and the kernel size must be specified as an integer. The width of the kernel is always the same size as the width of the time series in 1D CNNs. The kernel size that is specified is the length, illustrated as "k" in Figure 2.2. The kernel will always move in the same direction in 1D CNNs, from the beginning of the time series to the end. The pooling layer reduces the dimensionality of the activation maps by summarizing them using a pooling operation, usually taking the average or maximum value within a small window size. Finally, the fully-connected layer connects all neurons of the previous layer to all neurons in the next. The fully connected layer is implemented before the output layer that predicts the label of the input (O'Shea & Nash 2015). An example of a simple 1D CNN architecture can be seen in Figure 2.2.



Time	Speed (m/s)	Brake pedal position (%)	Temperature (C°)
-3.0	21.3	25.6	15.7
-2.8	21.5	25.6	15.7
-2.6	21.2	26.4	15.7
-2.4	20.6	26.5	15.7
-2.2	19.3	26.9	15.7
-2.0	19.0	31.4	15.7
-1.8	18.2	32.3	15.7
-1.6	18.2	26.4	15.7
-1.4	18.0	26.9	15.7
-1.2	17.4	27.5	15.7
-1.0	17.3	28.9	15.7

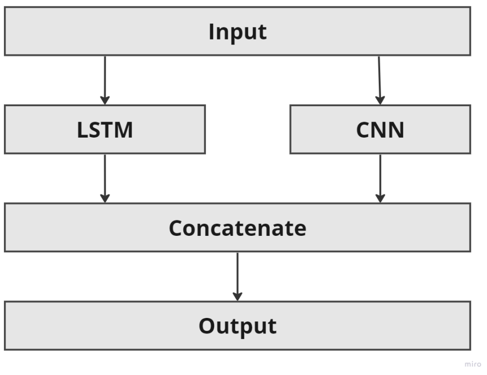
Figure 2.2: Sketch of the architecture of a simple 1D CNN with one convolutional layer consisting of four filters and a kernel size of three, represented as k in the figure. The input is time series data and the output is one of the two labels "hazardous" and "non-hazardous".

CNNs are mostly used for applications in image and speech recognition. However, in recent years the use of CNNs for both univariate and multivariate time series classification has been explored (LeCun et al. 1995, Zhao et al. 2017). When having multivariate time series, Zhao et al. (2017) suggested that the input layer consists of $N \times m$ neurons, where m represent the number of instances and N represent the number of features in each time series. Further, Zhao et al. (2017) showed that the number of filters, the size of the filters, and the size of the pooling layers can drastically affect the performance of the model.

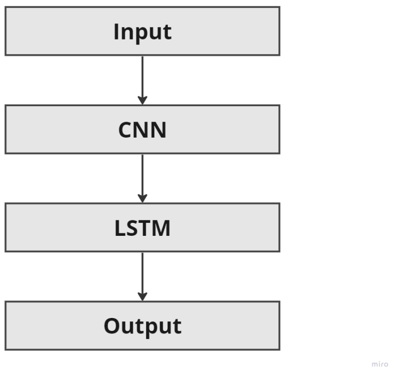
2.3 CNN-LSTM architecture

Osman & Hajij (2021) showed that both the CNN and LSTM models separately performed well when it came to predicting vehicle crashes. In order to improve the result even more, some researchers explored a combination of a CNN and an LSTM. Their results showed that a combined architecture improved the result of TSC tasks (Fullah Kamara et al. 2020, Karim et al. 2019). The advantage of the combined model was that the LSTM was better at learning based on both long- and short-term memory whereas the CNN was better at handling time-invariant features (Li et al. 2020).

In the literature, multiple different architectures of CNN-LSTM combined models have been used for TSC tasks. One possible architecture is to create a model where the input is processed by a CNN and an LSTM model simultaneously (Fullah Kamara et al. 2020, Karim et al. 2019). The architecture of the **Parallel CNN-LSTM** is illustrated in Figure 2.3 (a). Each model will get the time series as input and both of the outputs will be concatenated after each model has been performed. Subsequently, both values will be processed in an output layer consisting of a softmax function. This process will return a prediction of the classification (Fullah Kamara et al. 2020, Karim et al. 2019, Li et al. 2020).



(a)



(b)

Figure 2.3: Parallel and Sequential CNN-LSTM architecture.

Another approach is to apply the different models after each other, the **Sequential CNN-LSTM** is illustrated in Figure 2.3 (b). This architecture is a sequential model where the input first is processed by a CNN network and its output will be the input to the LSTM model. The output from the CNN maintains the same dimensions as the input, allowing it to serve as the input to the LSTM. The LSTM is followed by a fully connected layer before the output layer. By applying a CNN before an LSTM and not the other way around, the CNN can act as a feature extractor, identifying local patterns and high-level abstractions. The output from the CNN is then processed by an LSTM, which are known to be good at capturing temporal dependencies (He et al. 2022, Livieris et al. 2020, Kim & Cho 2019, Barzegar et al. 2020).

2.4 Evaluation Metrics

There are various methods available to evaluate the performance of supervised classification models. When using binary classification models, there are four different outcomes:

- **True Positive (TP)**: label is positive, predicted as positive.
- **False Negative (FN)**: label is positive, predicted as negative.
- **True Negative (TN)**: label is negative, predicted as negative.
- **False Positive (FP)**: label is negative, predicted as positive.

From this the *True Positive Rate (TPR)* and the *FPR* can be computed. These metrics are then used to produce the Receiver Operating Characteristics (ROC) curve at varying thresholds (Bradley 1997). In the ROC curve, the TPR is plotted on the y-axis and the FPR on the x-axis. Calculating the Area Under the ROC Curve (AUC) provides a way to measure the performance of the classification across all thresholds (Bradley 1997, Ferrer 2022). An AUC score of 1.0 indicates that the model is perfect at distinguishing between the two classes. The formulas for calculating the TPR and FPR for the ROC curve are shown below.

$$FPR = \frac{FP}{FP + TN}.$$

$$TPR = \frac{TP}{TP + FN}.$$

The aim of the thesis is to classify a situation as hazardous or not in hopes of notifying the driver in time. It is important that the model does not falsely notify the driver, therefore a low FPR rate is preferred. In contrast to the automatic braking function where the False Negative Rate (FNR) is more important considering no hazardous situations should be missed, the proposed model aims to notify the driver further in advance. A commonly used threshold for the FPR is 10%. This threshold will be used during the evaluation of the models. Another widely used evaluation metric for classification models is *accuracy*. The metric tends to be biased if the classes are unbalanced. Calculated by the positives and the negatives the formula is as follows.

$$Accuracy = \frac{TN + TP}{TP + FP + TN + FN}.$$

The F1-score is another popular metric for the evaluation of classification models and it represents the average of the *precision* and *recall* (Ferrer 2022). The F1-score can tell if there is a good balance between how often the predicted hazardous situations are actual hazardous situations (precision), and the correct predictions of actual hazardous situations (recall), where

$$Precision = \frac{TP}{TP + FP},$$

$$Recall = \frac{TP}{TP + FN},$$

$$F1 - score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}.$$

3 Methods

This chapter introduces the datasets and the pre-processing steps. In Section 3.1 the description of the data is presented. Data pre-processing steps are discussed in Section 3.2 which include the data cleaning, feature selection, transformation of data, and fine-tuning techniques. Lastly, in Section 3.3 the architectures of the machine learning models are presented.

3.1 Data Description

The recorded vehicle data can be described as multivariate time series data since there were multiple features varying over time. The features were of mixed data types consisting of both numerical and nominal categorical variables. The dataset was divided into two main parts, one which will be referred to as the hazardous driving dataset, and the other non-hazardous driving dataset. In addition, external weather data was added to the model. As mentioned previously, the hazardous driving dataset was collected from customer trucks, and the non-hazardous driving dataset was collected from development trucks. Since these two datasets differ in the way they were collected, subsections have been created to better describe them.

3.1.1 Hazardous Driving Dataset

The hazardous driving dataset consisted of short sequences of recorded Controller Area Network (CAN) data a few seconds around the time of a function intervention. Specifically, the sequences were 3 seconds long, with a frequency of 5 Hz. The information stored was about the truck's movement, for example, its speed, angle, and brake pedal position. Some information about the object that triggered the function intervention was also recorded. An object can be a car, truck, or pedestrian. Example of information stored about the object is the speed of the object, longitude position, and latitude position. The hazardous driving dataset was collected throughout 2022 from customer trucks registered in Europe. Each vehicle recorded roughly 90 features and the number of events in the hazardous driving dataset were tens of thousands. All events in the hazardous driving dataset were triggered by a function intervention and therefore were considered hazardous situations. As mentioned earlier, a hazardous situation is represented either by automatic braking of the vehicle and no crash, or automatic braking and a crash. Automatic braking includes both pre-brakes and full-brakes, and the distribution between them is unknown. When the driver does not react to a risk of collision, moderate braking will occur, namely the pre-braking. A full-brake, on the other hand, happens when the collision risk is not reduced and full emergency braking gets activated. All instances of automatic braking are considered hazardous situations.

3.1.2 Non-Hazardous Driving Dataset

In addition to the hazardous driving dataset, a dataset with continuous CAN data was used. The non-hazardous dataset was collected from a few development trucks registered in Europe consisting of tens of thousands of events. The data is recorded from development trucks, with the assumption that the dataset does not contain any hazardous situations. Therefore, all events in the dataset were regarded as non-hazardous situations. Similarly to the hazardous driving dataset, the data was collected in 2022. Three-second long snippets were extracted from each driving instance with a maximum of seven driving sessions from each truck-date combination, which resulted in a reduction in the number of events, more described in Section 3.2.2.

3.1.3 Weather Dataset

Furthermore, it was of interest to explore if information about the weather could enhance the accuracy of the predictions. Open-meteo is a free open-source API that includes data about temperature, precipitation, cloud cover, humidity, and wind speed. Data from open-meteo's Historical Weather API was extracted for every instance in the hazardous and non-hazardous driving dataset. The weather data was available every hour and had an average resolution of 11 km (Open-Meteo n.d.).

3.2 Data Pre-processing

Before the data was used in the machine learning models some pre-processing steps had been carried out. These steps included a few ways to make sure that the data was as meaningful as possible for the purpose of the thesis.

3.2.1 Combination of Different Data Sources

The first step was to combine data from different data sources. The data from the different datasets required matching columns, data types, and names. The hazardous, non-hazardous, and weather datasets were merged together into one dataset. In this step, the choice of the distribution for the labeled data was necessary, given that the non-hazardous and hazardous driving datasets were of different sizes. The non-hazardous driving dataset consisted of considerably fewer instances than the hazardous driving dataset. To avoid bias during the training of the models, half of the dataset consisted of "hazardous situations" and the other half of "non-hazardous situations". This was done by removing events in the hazardous driving dataset. The distribution of some features for each data label is illustrated in Figures 3.1, 3.2, and 3.3. As seen in Figure 3.3, the distribution of the geographical location differ

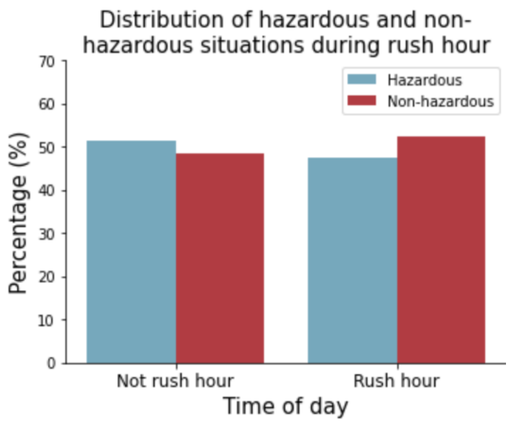


Figure 3.1: Distribution of the two classes between rush hour (5-9 & 16-21) and not rush hour.

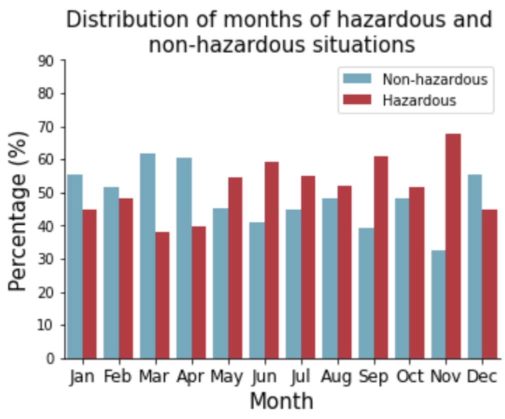


Figure 3.2: Distribution of the two classes between months.

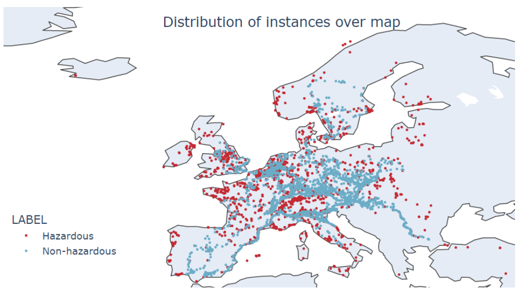


Figure 3.3: Geographical distribution of the two classes.

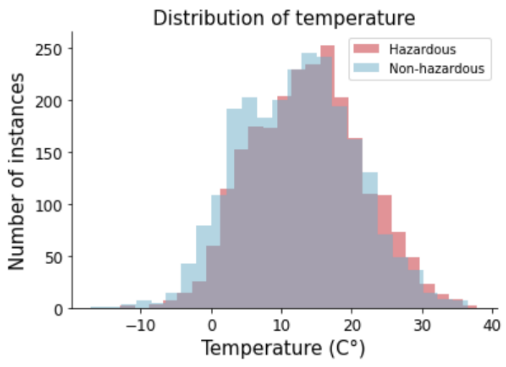
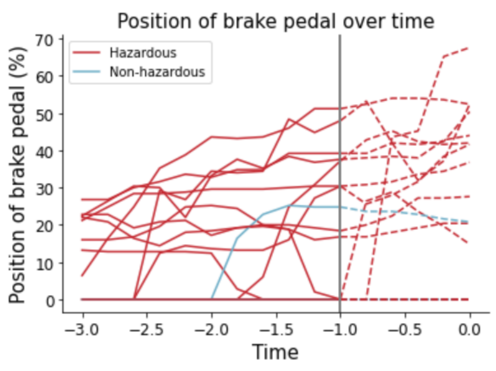


Figure 3.4: Distribution of temperature between the two classes.

between the hazardous and non-hazardous instances. When analyzing the weather features, a balanced distribution of the features across both classes can be seen. This suggests that the absence of some non-hazardous instances in some countries should not impact the models.

The second step was to choose the length of the time series. As mentioned previously, the hazardous driving situations only had three seconds of data prior to the function intervention, and in total 16 time logs, since the truck recorded data with a frequency of 5 Hz. An example of a time series can be seen in Figure 3.5. The second before the intervention occurs, time log -1.0 to 0.0 in Figure 3.5, a change in the pattern can be distinguished. As a consequence, there was a risk that the last second differed greatly compared to non-hazardous driving. The time logs between -1.0 and 0.0 were excluded to mitigate overfitting. Hence, each driving instance consisted of 11 time logs. In Figure 3.5, the dotted line is excluded from the time series. For the non-hazardous driving dataset, two second long snippets were randomly extracted to match the dimensions of the hazardous driving dataset. When extracting these snippets, a condition was applied to ensure that the truck was in motion and not stationary.



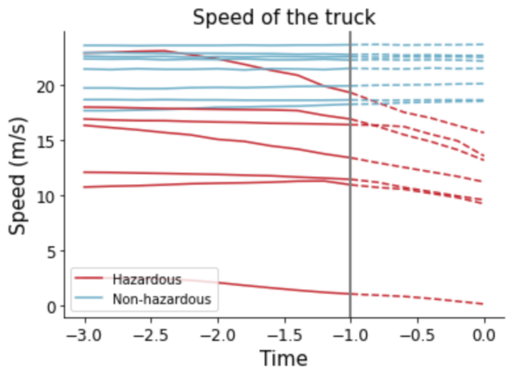


Figure 3.5: Line plots of brake pedal position and speed of the truck prior to the function intervention. At time log 0.0 the function intervention occurs. The last second (dotted lines) is excluded from the dataset.

3.2.2 Data Cleaning

When the dataset was combined into one source, some cleaning of the data for inconsistent values was necessary. This included investigating if the dataset contained incorrect or missing data. Instances where missing data occurred were excluded from the dataset. For all features, a range of valid values was identified, either from the information from the different sensors or based on domain knowledge. This range was used to exclude incorrect data. For instance, the valid range for brake pedal position is 0-100, any data points with values outside of this range were removed.

The next data cleaning step was to decide the size of the dataset. Having in mind that the non-hazardous driving dataset was collected from only a few development trucks in a limited geographical area, using all the data points would create underlying patterns and affect the generalization of the model in a negative way. Therefore, a maximum of seven driving sessions from each truck-date combination was used to help minimize underlying driving patterns from the same driving session. The outcome was 2426 data points from non-hazardous situations and 2402 data points from hazardous situations. In total, the final dataset consisted of 4828 instances, each with 11 time logs.

After cleaning, the final dataset was divided into a training and test set with an 80/20 split. The test set was set aside and not used until the final evaluation of the models. The remaining dataset was used during the training and optimization of the models as well as for the feature selection step.

3.2.3 Feature Selection

Having in mind that the features of the datasets included mixed data types, the standard correlation metrics do not work well. Therefore, a different approach was used. The feature selection process was only performed using the training set to avoid bias. The dataset included over 100 different features. The first exclusion of features was automatically done when combining the hazardous, non-hazardous, and weather datasets since some features in the hazardous driving dataset did not exist in the non-hazardous driving dataset. The second exclusion occurred when the distribution of the labels between the features was investigated. If the distribution was skewed because of the way the non-hazardous and hazardous driving datasets were collected, that feature was excluded. One example of a skewed feature is the collision warning feature which was never activated for the non-hazardous driving dataset. In addition, some features were always equal for both labels and over all time stamps. These features were excluded to minimize the dimensions of the dataset, thus making the training process for the models shorter. Further, when analyzing the features, it was found that some functions of the trucks were disabled when collecting the non-hazardous data. These features were removed in order to most accurately reflect real driving. Additionally, the GPS features were excluded from the models to mitigate overfitting and focus on the weather features.

3.2.4 Transformation of Data

The data needed to be transformed into a correct format so that it could be used as input for the models. The original structure of the dataset can be seen in Table 3.1. Each instance in the dataset had 11 rows, representing 11 different time logs, and the features were organized into columns. For the RF classifier, the input shape needed to be one-dimensional. The data was transformed into a dataframe where each row was a driving instance, and the columns were the features during the 11 different time logs, see Table 3.3. The unique ID that identified each driving instance was deleted in this step. The features were handled differently depending on if they were static or time-variant. The shape of the dataframe was therefore (3863, 151) where 3863 was the number of driving instances and 151 was the number of time-variant columns times the number of time logs plus the static features ($13 \times 11 + 8 = 151$). Static features were the ones about the weather and if it was rush hour or not. The static features had the same values throughout all time logs.

For the neural networks, the input shape needed to be three-dimensional. The data was transformed into an array that consisted of dataframes. Each dataframe represented one instance with the features as columns and the time logs as rows, see Table 3.2. Hence, the input shape was (3863, 11, 28) where 3863 was the number of instances, 11 was the number of different time logs and 28 was the number of features. The grouping of time logs was based on a unique ID that was deleted in this step to remove driver traceability. The numerical features were normalized using min-max scaling within the range of zero to one. To avoid bias, the mean and variance for the normalization were only calculated from the training set. For the non-binary categorical variables, one-hot encoding was performed. In one-hot encoding, the categorical features are transformed into a vector that only has binary values of zero and one. Initially, the feature "rush hour" was a temporal feature that included all hours. However, in line with the objective of the thesis, a transformation of this feature into the binary options "rush hour" and "not rush hour" was made. Rush hour was considered the times between 5:00-9:00 and 16:00-21:00 (Wikipedia 2023). The transformation aligned more closely with the research questions of the thesis and reduced the complexity of the feature space.

Table 3.1: Example of the original data structure before the transformation. Only one instance is shown and a subset of the features. The values for the features are dummy values.

Instance	Speed (m/s)	Brake pedal position (%)	Temperature (C°)	Time
1	21.3	25.6	15.7	-3.0
1	21.5	25.6	15.7	-2.8
1	21.2	26.4	15.7	-2.6
1	20.6	26.5	15.7	-2.4
1	19.3	26.9	15.7	-2.2
1	19.0	31.4	15.7	-2.0
1	18.2	32.3	15.7	-1.8
1	18.2	26.4	15.7	-1.6
1	18.0	26.9	15.7	-1.4
1	17.4	27.5	15.7	-1.2
1	17.3	28.9	15.7	-1.0

Table 3.2: Example of the data structure for the DLs. Three instances are shown and a subset of the features. The last and first time logs are shown. The values for the features are dummy values.

Instance	Dataframe			
1	Speed	Brake pedal position	Temperature	Time
	23.6	23.4	15.7	-3.0
	...	...	...	...
	21.7	27.8	15.7	-1.0
2	Speed	Brake pedal position	Temperature	Time
	34.9	13.6	24.3	-3.0
	...	...	...	...
	27.5	14.9	24.3	-1.0
3	Speed	Brake pedal position	Temperature	Time
	17.9	0.0	2.9	-3.0
	...	...	...	...
	16.6	0.0	2.9	-1.0

Table 3.3: Example of the data structure for the RF classifier. Three instances are shown and only a subset of the features. All time logs are not included in the table and the values for the features are dummy values.

	Time -3.0			Time -2.8			...	Time -1.0		
Instance	Speed	Brake pedal position	Temp	Speed	Brake pedal position	Temp	...	Speed	Brake pedal position	Temp
1	21.3	25.8	15.7	21.3	25.7	15.7	...	17.3	29.6	15.7
2	28.7	23.3	34.4	28.7	24.3	34.4	...	24.5	30.2	34.4
3	18.2	17.6	-5.3	18.2	17.6	-5.3	...	16.7	24.8	-5.3

3.2.5 Fine-tuning

10-fold cross validation was used for all five models when fine-tuning. 10-fold cross validation splits the dataset into ten sub-groups and in the first fold trains on the first nine of the sub-groups and evaluates on the tenth. In the second fold, training is done on another subset of nine sub-groups and evaluated on the last sub-group. This is then repeated 10 times and the average of the results of each fold is used to evaluate the performance of the model.

Random search was used during hyperparameter tuning. Random search is a method that searches randomly through a specified range of hyperparameters trying to find the combination that yields the highest accuracy. To find the optimal combination, 1000 combinations of hyperparameters were tried out. When the random search was finished, the optimal values for the hyperparameters were used to re-train the models using the whole training set. The models were then evaluated on the test set.

3.2.6 Adding Noise

To address the research question (c) - *Can the classifier improve its performance when noisy sensor data is introduced?* - noise was introduced into both the training and test sets. It was of interest to examine this setting because, in real-world scenarios, sensors can malfunction or deteriorate over time, leading to imprecise values. Investigating if the noisy data could obtain a high accuracy for classifying hazardous and non-hazardous situations is what this scenario reflects. For the numerical features in the training and test sets, gaussian noise was added with a mean of 0 and a standard deviation of 0.01. The tuning process involved adjusting the value of the standard deviation, which is a hyperparameter. Regarding the categorical features, noise was introduced by flipping the categories with a certain probability. The probability was set to 40%, meaning that there was a 40% chance of flipping the label. In a binary case this would mean that there was a 60% chance of keeping its original label and a 40% chance of flipping to the other label. In a multi-label scenario, if a flip occurred, the remaining labels had an equal probability distribution. The probability value was tuned with different values, considering it was a hyperparameter.

3.3 Model Architecture

As mentioned previously, an RF classifier was used as one of the baseline models. In addition to the simpler machine learning model, a CNN and an LSTM were used separately for comparison to the hybrid model. Therefore, the baseline models were an RF, a CNN, and an LSTM.

3.3.1 Baseline models

As mentioned previously, a random search with 1000 combinations of hyperparameters was performed. The input shape for the **RF classifier** was (3863×151), where 3863 was the number of driving instances and 151 was the number of features. The number of trees in the forest was 120 with the maximum amount of depth in each tree being 10. The weights between the two classes were balanced for each sample for every tree. The number of instances required to split a node was set to 10. The limit of how many features to consider when looking for the optimal split was the fraction 0.1.

After performing cross validation to optimize the **CNN**, a three-layer network was implemented. The input and output layers were not included when counting the layers. The CNN consisted of two convolutional layers and one fully connected layer. The input shape of the time series was ($3863 \times 11 \times 28$), where 3863 was the number of instances, 11 was the number of time logs, and 28 was the number of features. The learning rate was 0.001 and the batch size was 32. The *Adam* optimizer was used during training. The training ran for 100 epochs and the chosen hyperparameters from the random search 10-fold cross validation can be seen in Table 3.4.

Table 3.4: Hyperparameter tuning results for the CNN. The column *Range* represents the values randomly tested whereas the column *Value* presents the final value for each hyperparameter.

Hyperparameter	**Range**	**Value**
Number of convolutional layers	2, 3	2
Filter size	32, 64, 128	64
Kernel size	3, 5, 7	3
Pool size	2, 3, 4	2
Number of nodes for dense layer	16, 32, 64	64
Dropout	0.1, 0.2, 0.3	0.2
Learning rate	0.0001, 0.001, 0.01	0.001
Batch size	16, 32, 64, 128	32
Epochs	50, 100, 150	100
Patience for early stopping	10	10

The **LSTM** model consisted of two recurrent LSTM layers. The input shape of the LSTM model was $(3863 \times 11 \times 28)$ and the LSTM layer used *tanh* as the activation function and *sigmoid* as the recurrent activation function. Both of the hidden layers consisted of 64 units and after both layers, a dropout was applied. The dropout was initially 0.2 and for each layer, the value was reduced by 0.1. The network was compiled with the *Adam* optimizer and a learning rate of 0.01. The best result for the LSTM was reached with a batch size of 16 and 100 epochs. The chosen hyperparameter values from the cross validation can be seen in Table 3.5.

3.3.2 CNN-LSTM

The input of both of the hybrid CNN-LSTM models was a $(3863 \times 11 \times 28)$ matrix, similar to the individual CNN and LSTM models. The **parallel CNN-LSTM** was implemented using the Keras functional API, which is a way to create more complex models that can be processed in parallel. With this API models with multiple inputs, outputs, and shared layers can be created. In the parallel CNN-LSTM, the CNN model consisted of two convolutional and pooling layers followed by a flattening layer. The last step of the CNN model is a Dense layer with 64 nodes. The LSTM

Table 3.5: Hyperparameter tuning results for the LSTM. The column *Range* represents the values randomly tested whereas the column *Value* presents the final value for each hyperparameter.

Hyperparameter	**Range**	**Value**
Number of layers	1, 2, 3	2
Number of nodes in LSTM	16, 32, 64	64
Initializer	glorot uniform, orthogonal, truncated normal	orthogonal
Dropout	0.2, 0.3, 0.4, 0.5	0.2
Learning rate	0.0001, 0.001, 0.01	0.01
Batch size	16, 32, 64, 128	16
Epochs	50, 100, 150	100
Patience for early stopping	10	10

Table 3.6: Hyperparameter tuning results for the Parallel and Sequential CNN-LSTM. The column *Range* represents the values randomly tested whereas columns *Parallel* and *Sequential* present the final value for each hyperparameter.

Hyperparameter	**Range**	**Parallel**	**Sequential**
Number of LSTM layers	1, 2, 3	1	2
Number of nodes for LSTM	16, 32, 64	32	16
Initializer	glorot uniform, orthogonal, truncated normal	orthogonal	orthogonal
Dropout LSTM	0.2, 0.3, 0.4, 0.5	0.3	0.5
Number of convolutional layers	1, 2, 3	2	3
CNN filter size	32, 64, 128	32	64
CNN kernel size	3, 5, 7	7	5
CNN pool size	2, 3, 4	4	2
CNN dropout	0.1, 0.2, 0.3	0.3	0.1
CNN nodes for dense layer	32, 64, 128	64	NA
Number of nodes for dense layer	16, 32, 64, 128	64	32
Learning rate	0.0001, 0.001, 0.01	0.001	0.001
Batch size	16, 32, 64, 128	32	16
Epochs	50, 100, 150	150	100
Patience for early stopping	10	10	10

model was composed of one LSTM layer with a dropout of 0.3. The output from the CNN and LSTM was concatenated and processed through a Dense layer with ReLu function activation before the output layer.

The **Sequential CNN-LSTM** model was implemented using the Sequential class in Keras. The model architecture consisted of three convolutional and pooling layers, followed by two LSTM layers. After each LSTM layer, dropout layers were implemented to help mitigate overfitting. No fully-connected layer was implemented after the pooling layers in the CNN in order to have the output from the CNN layers as input to the LSTM layers.

Both the parallel and sequential CNN-LSTM models were tuned by using random search on the range displayed in Table 3.6. The hyperparameter values were chosen from a 10-fold cross validation shown in the last two columns in the table. The final architectures of the sequential and parallel CNN-LSTM are visualized in Appendix B.

4
Results

This chapter presents the results of the models. The performance of the models was evaluated by testing them on data consisting of both hazardous and non-hazardous situations. To evaluate the models an 80/20 train/test split was used. Section 4.1 presents and compares the results from all five models. In Section 4.2, the results from all models when the weather data is excluded from the dataset are presented. Lastly, in Section 4.3, the results from all models when the dataset includes noise are presented.

4.1 Result from all models

In Table 4.1 the result from training the five models are presented. The result for the training was retrieved by training the models with 10-fold cross validation. The metrics in the table under column **Train** are the average over all folds. The results when evaluating the models using the test set are seen under column **Test** in the table.

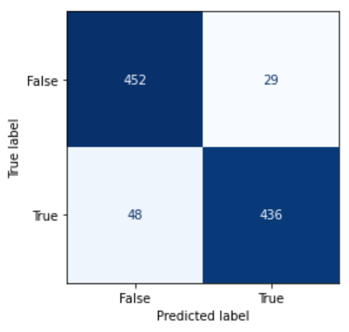
As presented in Table 4.1, all models except for the LSTM attained a training set accuracy of around 92%. When looking at the result from the test set, RF achieved the highest accuracy score of 92% which was less than half a percentage point lower than the training result. Moreover, the F1-score and AUC for the RF resulted in a minor decrease in performance when evaluated on the test set. As for the CNN, its testing set accuracy resulted in 79.5%, which was a decrease of 12.1 percentage points compared to the training set accuracy. The F1-score and AUC metrics showed similar behavior for the CNN on the test set. The parallel CNN-LSTM model attained a testing set accuracy of 81.2% and an F1-score of 77.6%. The AUC score for the testing set resulted in a value of 81.3%. Whereas the sequential CNN-LSTM obtained a test accuracy and AUC score of 67% and an F1-score of 73.6%. The average training accuracy for the LSTM resulted in a value of 82.7%, an F1-score of 81.7%, and an AUC of 91.5%. As for the test set, the LSTM attained an accuracy of 68.5%, an F1-score of 55.3%, and an AUC of 68.6%.

As shown in Table 4.1, both hybrid CNN-LSTMs seem to have similar scores for the training data as for the other models. However, when comparing the test result among the models, the LSTM and sequential CNN-LSTM perform the poorest. The RF performs the best with similar scores on the test set as the training set which is an indication of neither under- or overfitting.

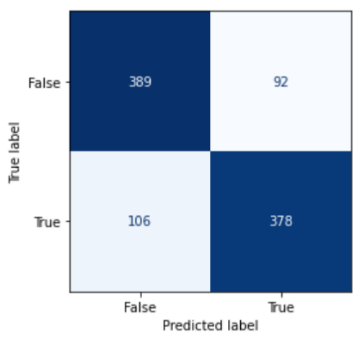
Table 4.1: Accuracy, F1-score, and AUC for training and test set for each model.

	Train			Test			
Model	**Accuracy**	**F1-Score**	**AUC**	**Accuracy**	**F1-Score**	**AUC**	**FPR**
RF	0.924	0.922	0.981	0.920	0.919	0.920	0.060
CNN	0.916	0.912	0.961	0.795	0.792	0.795	0.191
LSTM	0.827	0.817	0.915	0.685	0.553	0.686	0.017
pCNN-LSTM	0.924	0.923	0.924	0.812	0.776	0.813	0.021
sCNN-LSTM	0.925	0.922	0.968	0.670	0.736	0.670	0.575

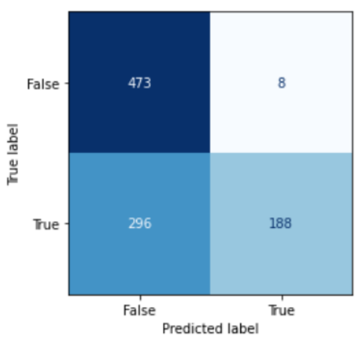
Moreover, the FPR scores are shown in Table 4.1. The FPR was of interest considering an optimal model would not falsely notify the driver that a hazardous driving situation will occur. When evaluating the performance of the models based on the FPR, a value below 10% was considered good. As shown in the table the RF, LSTM, and the parallel CNN-LSTM managed a FPR below 10%. The FPR scores were computed based on the FPs and the TNs, which can be seen in the first row of the confusion matrices in Figure 4.1. An interesting result that can be seen in Figure 4.1 (c) is that the LSTM had the lowest FPR but achieved the lowest accuracy since it predicted many instances to be false. The LSTM and sequential CNN-LSTM achieved similar accuracies on the test set but different FPRs, this is due to the fact that the LSTM predicts many instances as non-hazardous situations and the sequential CNN-LSTM predicts many instances as hazardous situations.



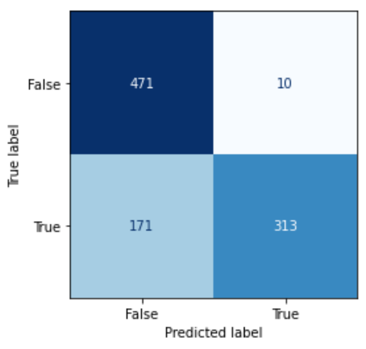
(a) RF



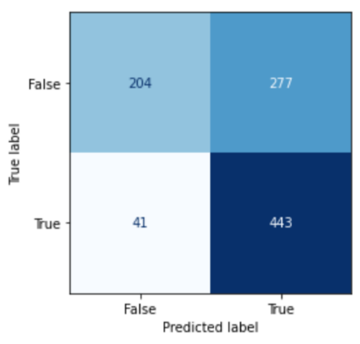
(b) CNN



(c) LSTM



(d) Parallel CNN-LSTM



(e) Sequential CNN-LSTM

Figure 4.1: Confusion matrices for each model.

Table 4.2: Accuracy, F1-score, and AUC for training and test set for each model without weather data.

	Train			Test			
Model	**Accuracy**	**F1-Score**	**AUC**	**Accuracy**	**F1-Score**	**AUC**	**FPR**
RF	0.911	0.910	0.976	0.733	0.785	0.732	0.511
CNN	0.936	0.932	0.973	0.708	0.737	0.707	0.401
LSTM	0.850	0.850	0.934	0.792	0.778	0.792	0.143
pCNN-LSTM	0.932	0.928	0.932	0.822	0.792	0.822	0.033
sCNN-LSTM	0.924	0.922	0.970	0.717	0.613	0.718	0.010

4.2 Result without weather data

The results from all models, trained and tested on data excluding the weather features, are presented in Table 4.2. The training set metrics were obtained through 10-fold cross validation using the hyperparameters that resulted in the highest validation accuracy during the random search. As shown in the table, the RF obtained a training set accuracy of 91.1% and a test set accuracy of 73.3%, a decrease of almost 18 percentage points. Similar behavior can be seen in the other models. The model with the highest test set accuracy was the parallel CNN-LSTM with a FPR of 3.3%. Out of all models, only the parallel and the sequential CNN-LSTM had an FPR below 10%. The confusion matrices used to calculate the FPRs can be found in Appendix A.

When comparing this result to the result including weather data in Table 4.1, the training and test set results are quite similar between the models. The test result of the RF seems to be the most affected by the removal of the weather data, where the accuracy for the RF decreased from 91.6% to 73.3%. Similarly, the accuracy of CNN decreased by almost 9 percentage points when the weather data was excluded. In contrast, the accuracy for LSTM increased from 68.5% to 79.2%. The result from the parallel and sequential CNN-LSTM was relatively stable regardless of the removal of weather data.

4.3 Result when adding noise

In Table 4.3 the results from all five models when including noise to the training and test sets are presented. The metrics from the training set were obtained from 10-fold cross validation. The hyperparameters that achieved the highest validation accuracy in the random search for the individual models were used for the cross validation. In the table, we can see that the RF has the highest training set accuracy of almost 90%. As for the test set accuracies, the RF obtained the highest score, followed by the parallel CNN. All models had lower FPRs, with only the RF achieving an FPR below 10%. The confusion matrices used to calculate the FPRs can be found in Appendix A.

Table 4.3: Accuracy, F1-score, and AUC for noisy training and test sets for each model.

	Train			Test			
Model	**Accuracy**	**F1-Score**	**AUC**	**Accuracy**	**F1-Score**	**AUC**	**FPR**
RF	0.893	0.888	0.963	0.896	0.894	0.896	0.077
CNN	0.854	0.851	0.929	0.839	0.843	0.893	0.179
LSTM	0.837	0.837	0.921	0.809	0.815	0.809	0.216
pCNN-LSTM	0.875	0.872	0.876	0.815	0.807	0.816	0.135
sCNN-LSTM	0.859	0.855	0.929	0.825	0.830	0.825	0.202

From the results, we can see that when noise is added the difference between the training and testing result is smaller compared to the previous tests with and without weather data. Compared to the results in Table 4.1 and Table 4.2, the scoring values are more even among the models. Further, we can observe that all models except the RF performed better when noise was added.

5 Discussion

This chapter will include discussions around the performance of the models, the choice of method, and its limitations. In Section 5.1 a summarization of the results obtained in Chapter 4 will be presented and commented on. In Section 5.2 a discussion about the different models will be presented, as well as some limitations of the thesis.

5.1 Model performance

The following section analyzes and discusses the performance of the models and their relation to the research questions stated at the beginning of this thesis. The research questions address how a CNN-LSTM model performs compared to the two separate models and if the models improve when including data about the weather. The last research question addresses how noisy sensor data affects the performance of the models. As a justification for using Deep Learning (DL) models to solve the aim of the thesis, an RF model was used for comparison.

5.1.1 Complete dataset

The result from the complete data set showed that the RF outperformed the rest of the models when deployed on the test set. Moreover, RF is the only model that had similar scoring values on the training and test sets compared to the other models that decreased in performance when deployed on the test set. The DL models had high scoring values on the training set, around 90%, however the test set scoring values were close to 70% for the LSTM and sequential CNN-LSTM, and close to 80% for the CNN and parallel CNN-LSTM. The decrease in performance from the training set to the test set is a sign of overfitting. All DL models used dropout in between layers and early stopping to omit overfitting as much as possible, but looking at the results it seems that the regularization techniques used might not have been enough. Another reason that might have caused overfitting is the complexity of the DL models. A more complex model is prone to learning the training data too well, resulting in lower test set results. A third reason that could be the cause of overfitting is the size of the datasets. DL models are data hungry and can easily overfit if the dataset is small, which is one of the limitations of this thesis. A combination of the three reasons mentioned is what we believe have affected the results from the DL models on the test sets.

5.1.2 Without weather data

When excluding the weather data, we can see a similar behavior for all models when it comes to overfitting. Without the weather data, even the RF overfits considering the test set evaluation metrics are much lower than the training set metrics. Since each decision tree in the RF only considers a subset of the features the exclusion of the weather data might have affected the model to draw important connections and patterns. Some DL models performed slightly worse without the weather data and some better. This can be an effect of the natural randomness DL models have when initiating the weights. For the RF and CNN, the accuracies were lower when excluding weather data and the FPRs were higher. With this combination, we can draw the conclusion that the two models perform worse without weather data. For the LSTM, the model achieved an average of 10.7 percentage points higher accuracy when excluding the weather data. On the other hand, the FPR was above 10%, meaning that the model predicts many instances to be hazardous even though they are not. In a scenario where the model is implemented in a vehicle in order to notify the driver about the situation being hazardous, a model that has a lower FPR would be preferable. However, when selecting the optimal model the metrics need to be weighted differently depending on the context.

When comparing the results from Table 4.1 and Table 4.2, we can observe that the test results of the parallel and sequential CNN-LSTMs are not fluctuating as much as the other models. This might be an indicator that the hybrid models are better at finding patterns without the need for additional weather data. From the results, we can draw the conclusion that the RF, CNN, and LSTM perform worse when excluding weather data, however, we cannot draw any conclusion about how it affects the hybrid models.

5.1.3 Adding noise

When noise was added to the datasets the difference between training and testing scores was reduced for all models. The training results were slightly lower when using noise compared to using the complete dataset and excluding the weather features. On the other hand, the testing results increased for all models except for the RF. Further, the training and test results for all models were similar, which is a sign that the models neither under- or overfitted the training data. The RF outperformed the DL models, similar to the results from the complete data set. For the CNN and sequential CNN-LSTM the accuracies were higher and the FPRs were lower, which is an indication that the models performed better when noise was added. The LSTM and parallel CNN-LSTM had higher accuracies but also higher FPRs. Despite the fact that the RF resulted in superior performance compared to all other models, it is noteworthy that the DL models achieved higher scores as well when noise was added. The conclusion we can draw is that all DL models perform better and are more robust when noise is added to the datasets when looking at the accuracy, F1-score, and AUC.

5.2 Method discussion

The hybrid CNN-LSTM has been proven by previous studies to work better for sequential data compared to the separate models, as discussed in Chapter 2. Most of the previous research has compared their hybrid CNN-LSTM with various other models, such as DL models, ensemble algorithms, and logistic regression. To achieve the goal of the thesis, the performance of a parallel and sequential CNN-LSTM have been evaluated and compared to an RF, a CNN, and an LSTM. Unlike previous studies within crash prediction, we are comparing our DL models with an RF. The RF is known to be a good ensemble algorithm that does not require as much data as the DL models. As mentioned earlier, the RF turned out to be a better model for the data in this specific project with its limitation. However, if the models were trained on a larger dataset the results might have been different. Nevertheless, it is important to consider that the DL models require more complex tuning and greater computational resources. The initial hypothesis was that the hybrid CNN-LSTMs would outperform the RF, which is why we chose it as one of our baseline models. Therefore, including simpler baseline models, like a logistic regression model or a decision tree, might have added value to the results of the thesis. Comparing DL models with simple baseline models, unlike RF, might be a more fair comparison considering the limitations of this study. Despite the fact that the RF outperformed our proposed CNN-LSTMs, the simplicity of the RF should not be regarded as a disadvantage. One advantage is the interpretability of an RF, which are easier to understand and are more transparent in how the predictions are made, compared to DL models.

The hazardous dataset includes both pre-brakes and full-brakes and, as mentioned previously, the pre-brake is when the vehicle slightly brakes automatically and a full-brake is when full braking power gets activated in order to bring the vehicle to a stop. Considering the pre-braking function is used for alerting the driver to react to a situation it can be discussed if it should be considered a hazardous situation or not. Depending on the driving style of the driver, the pre-braking will occur more or less often. One driver may prefer to keep a larger distance from other cars whereas another driver does not and most likely would activate the pre-brake more often than the first driver. In this context, some of the pre-brakes that were activated may not necessarily be considered hazardous, they are just that driver's driving style. Therefore, it is hard to define a hazardous situation since it is subjective to each driver. Including pre-brakes would mean that some situations we want to classify as hazardous may not actually be hazardous. Here we needed to consider the trade-off between including pre-brakes or not when training the models. Including pre-brakes means that there is a risk that some of the instances are not actually hazardous whereas others are. This would also mean that the training data is larger for the models. On the other hand, by only considering full-brakes when training the models we might eliminate the risk of including non-hazardous instances, however, we might miss some situations where the driver manages to avoid the full-brake by intervening. There are pros and cons to both decisions and therefore it might be of interest to investigate both options when drawing any conclusion.

5.2.1 Limitations

There are a few limitations in the thesis that have influenced the results of the chosen models. The first one is the dataset itself. As mentioned previously, the hazardous dataset consists of data from customer trucks, whereas the data from the non-hazardous dataset was from development trucks. This results in less variability in terms of drivers and vehicles for the non-hazardous dataset. This limitation was not considered when choosing the models. As known, DL models need lots of training data in order to not overfit. In the preprocessing phase, a trade-off between a larger dataset with underlying patterns or a smaller dataset with fewer underlying patterns was made. Underlying patterns, in this context, would be that many instances were from the same day and truck, and presumably from the same driver, which would result in similar driving behaviors among the instances. The choice of reducing the dataset was made in order to avoid the models from overfitting the data sample.

Since the training and test set included instances where the same driver and vehicle were used, we believe that including all data points could have potentially improved the results. However, evaluating the models on a test set where the non-hazardous instances are not from development trucks the performance is not guaranteed. Therefore, the assumption we are making is that the generalizability of the models is poor. The optimal case would be to train the models on data from customer trucks, where the hazardous and non-hazardous are both from real-life driving, unlike in our case.

Another limitation is the fact that the data was collected in Europe, with the underlying assumption that the vehicles were only driven within Europe. Since the GPS location was not included in the parameters for the models, the location itself was not considered to be a limitation of the models. However, there are other factors connected to the location that may affect the models, such as driving behavior, road conditions, and the variation in weather. The purpose of the models was not to be used outside of Europe, which is the reason for only including training instances registered in Europe. Nevertheless, driving in countries within Europe differ and including a more even spread among the countries can combat any bias and help the generalizability of the models.

6
Conclusion

Road traffic accidents can be prevented by using ADAS when a hazardous driving situation occurs. The hypothesis of the thesis was that by using vehicle data together with external weather data, trends and patterns can be identified in order to classify a certain situation as hazardous or not. Finding these patterns early on would be an alternative way of increasing road safety. Motivated by previous studies that have used and achieved high performance on the hybrid CNN-LSTM model for sequential data, the thesis investigated if the hybrid model could be applied in accident prediction using time series data. Two different architectures of the CNN-LSTM were implemented, one parallel and one sequential. These were compared to an RF classifier, a CNN, and an LSTM. The models were trained and evaluated on the complete dataset. Furthermore, the models were evaluated by excluding the weather features in both training and testing. Lastly, the performance of the models was explored when noise was added to the data.

From the results, a tendency to overfit can be observed for all DL models, even though they obtained an accuracy close to 80%. The models that were overfitting on the complete dataset were improved when noise was added. However, RF outperformed the rest of the models in two out of three tests. The best performance achieved by the RF classifier was on the complete dataset with an accuracy of 92%. By revisiting the three research questions posed at the beginning of the thesis, the parallel CNN-LSTM performed better than the CNN and LSTM in two out of three test cases. However, the CNN and parallel CNN-LSTM achieved comparable levels of accuracy during the testing phase. The LSTM performed worst in all cases, achieving the lowest accuracy. Further, the RF, CNN, and LSTM performed worse when excluding the weather features, however, no conclusion can be drawn about how the weather features affect the hybrid models. This is because the hybrid models achieved a similar performance with and without weather features. Lastly, all models except the RF performed better on the test sets when including noise in the training and testing of the models.

The result challenges the hypothesis that a hybrid CNN-LSTM is the most suitable model for predicting hazardous driving situations using recorded vehicle data together with external weather data. One important factor that might have affected the result is the insufficiency of real-world data for non-hazardous situations. Even though the best performing model achieved an accuracy of 92%, it is not recommended to apply the model in a real-world scenario at this stage, as it needs to be trained and evaluated on a larger dataset from real-life driving.

6.1 Future Work

As discussed in previous sections, testing the model on an out-of-sample dataset where the data comes from customer trucks would be the first step to evaluating the generalizability of the models. If the performance of the model after testing it on an out-of-sample test set does not correspond to the one in the thesis, including customer data in the training of the model can be an option. Another interesting direction to take as a continuation of the thesis is to only include full-braking in the hazardous dataset during training. By doing so, any situations that triggered a function intervention but were not actually considered to be hazardous would be eliminated.

Another direction for future work is to apply the models on longer time series in order to predict a hazardous driving situation further in advance. This can be attainable by collecting longer segments of vehicle data. In this thesis, some features had to be discarded in the making of the models since they were closely connected to an intervention. If longer segments are included these features can be included which can help the performance of the model.

Bibliography

Abidogun, O. A. (2005), Data mining, fraud detection and mobile telecommunications: call pattern analysis with unsupervised neural networks.

Barzegar, R., Aalami, M. T. & Adamowski, J. (2020), ‘Short-term water quality variable prediction using a hybrid cnn–lstm deep learning model’, *Stochastic Environmental Research and Risk Assessment* **34**(2), 415–433.

Bengio, Y., Simard, P. & Frasconi, P. (1994), ‘Learning long-term dependencies with gradient descent is difficult’, *IEEE Transactions on Neural Networks* **5**(2), 157–166.

Bradley, A. P. (1997), ‘The use of the area under the roc curve in the evaluation of machine learning algorithms’, *Pattern recognition* **30**(7), 1145–1159.

Breiman, L. (2001), ‘Random forests’, *Machine learning* **45**, 5–32.

Centers for Disease Control and Prevention (2020). Accessed: 2022-12-10.
URL: *https://www.cdc.gov/injury/features/global-road-safety/index.html*

Chen, H., Cao, L. & Logan, D. B. (2012), ‘Analysis of risk factors affecting the severity of intersection crashes by logistic regression’, *Traffic Injury Prevention* **13**(3), 300–307. PMID: 22607253.
URL: *https://doi.org/10.1080/15389588.2011.653841*

Ferrer, L. (2022), ‘Analysis and comparison of classification metrics’.

Fu, X., Meng, H., Wang, X., Yang, H. & Wang, J. (2022), ‘A hybrid neural network for driving behavior risk prediction based on distracted driving behavior data’, *PLOS ONE* **17**(1), 1–16.
URL: *https://doi.org/10.1371/journal.pone.0263030*

Fullah Kamara, A., Chen, E., Liu, Q. & Pan, Z. (2020), ‘Combining contextual neural networks for time series classification’, *Neurocomputing* **384**, 57–66.
URL: *https://www.sciencedirect.com/science/article/pii/S0925231219316364*

Gers, F. (2001), Long Short-Term Memory in Recurrent Neural Networks, PhD thesis, Ecole Polytechnique Federale de Lausanne.

He, Y., Lv, J., Liu, H. & Tang, T. (2022), ‘Toward the trajectory predictor for automatic train operation system using cnnndash;lstm network’, *Actuators* **11**(9).
URL: *https://www.mdpi.com/2076-0825/11/9/247*

Hochreiter, S. & Schmidhuber, J. (1997), ‘Long short-term memory’, *Neural Computation* **9**(8), 1735–1780.

Jovanis, P. P. & Chang, H.-L. (1986), ‘Modeling the relationship of accidents to miles traveled’, *Transportation Research Record* .

Karim, F., Majumdar, S., Darabi, H. & Harford, S. (2019), ‘Multivariate lstm-fcns for time series classification’, *Neural Networks* **116**, 237–245.
URL: *https://www.sciencedirect.com/science/article/pii/S0893608019301200*

Kashifi, M. T., Al-Sghan, I. Y., Rahman, S. M. & Al-Ahmadi, H. M. (2022), ‘Spatiotemporal grid-based crash prediction—application of a transparent deep hybrid modeling framework’, *Neural Comput. Appl.* **34**(23), 20655–20669.
URL: *https://doi.org/10.1007/s00521-022-07511-y*

Kim, T.-Y. & Cho, S.-B. (2019), ‘Predicting residential energy consumption using cnn-lstm neural networks’, *Energy* **182**, 72–81.

LeCun, Y., Bengio, Y. et al. (1995), ‘Convolutional networks for images, speech, and time series’, *The handbook of brain theory and neural networks* **3361**(10), 1995.

Li, P., Abdel-Aty, M. & Yuan, J. (2020), ‘Real-time crash risk prediction on arterials based on lstm-cnn’, *Accident Analysis Prevention* **135**, 105371.
URL: *https://www.sciencedirect.com/science/article/pii/S0001457519311108*

Livieris, I. E., Pintelas, E. & Pintelas, P. (2020), ‘A cnn–lstm model for gold price time-series forecasting’, *Neural computing and applications* **32**, 17351–17360.

Ma, X., Tao, Z., Wang, Y., Yu, H. & Wang, Y. (2015), ‘Long short-term memory neural network for traffic speed prediction using remote microwave sensor data’, *Transportation Research Part C: Emerging Technologies* **54**, 187–197.
URL: *https://www.sciencedirect.com/science/article/pii/S0968090X15000935*

Open-Meteo (n.d.), ‘Historical weather api’. Accessed: 2023-03-09.
URL: *https://open-meteo.com/en/docs/historical-weather-api*

O’Shea, K. & Nash, R. (2015), ‘An introduction to convolutional neural networks’.
URL: *https://arxiv.org/abs/1511.08458*

Osman, O. A. & Hajij, M. (2021), ‘Development and comparative analysis of advanced deep learning techniques for crash prediction in advanced driver support systems’, *Transportation Research Record* **2675**(12), 730–740.
URL: *https://doi.org/10.1177/03611981211031220*

Tigadi, A., Gujanatti, R., Gonchi, A. & Klemsscet, B. (2016), ‘Advanced driver assistance systems’, *International Journal of Engineering Research and General Science* **4**(3), 151–158.

Wikipedia (2023), ‘Rush hour’. Accessed: April 27, 2023.
URL: *https://en.wikipedia.org/wiki/Rush_hour*

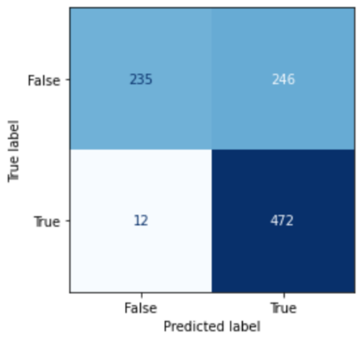
Yuan, Z., Zhou, X. & Yang, T. (2018), Hetero-convlstm: A deep learning approach to traffic accident prediction on heterogeneous spatio-temporal data, *in* ‘Proceedings

of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining', KDD '18, Association for Computing Machinery, New York, NY, USA, p. 984–992.
URL: *https://doi.org/10.1145/3219819.3219922*

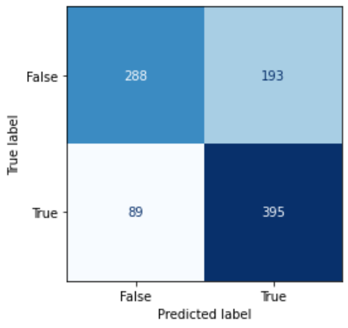
Zhao, B., Lu, H., Chen, S., Liu, J. & Wu, D. (2017), 'Convolutional neural networks for time series classification', *Journal of Systems Engineering and Electronics* **28**(1), 162–169.

A

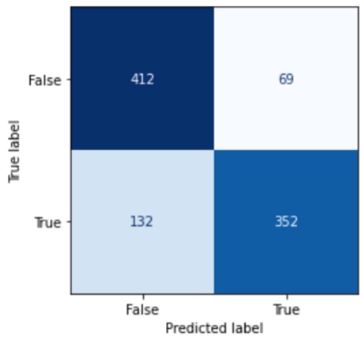
Appendix 1



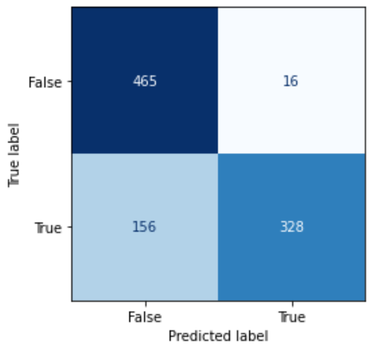
(a) RF



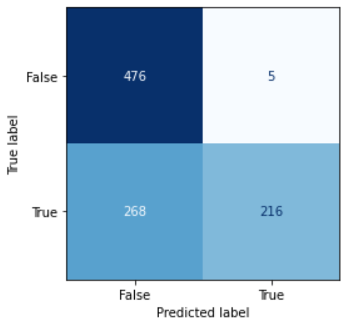
(b) CNN



(c) LSTM

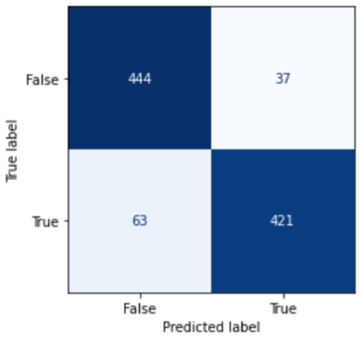


(d) Parallel CNN-LSTM

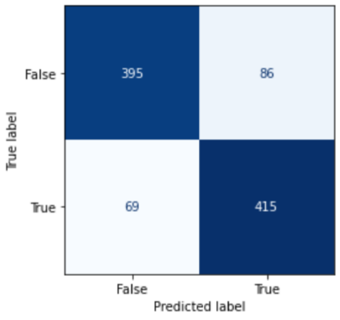


(e) Sequential CNN-LSTM

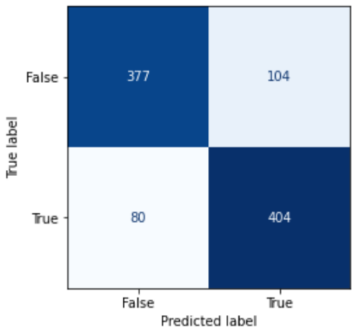
Figure A.1: Confusion matrices for each model without weather data.



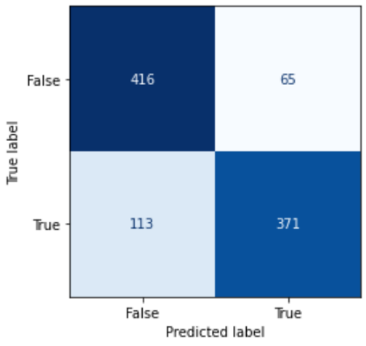
(a) RF



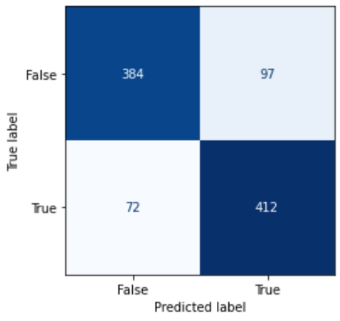
(b) CNN



(c) LSTM



(d) Parallel CNN-LSTM

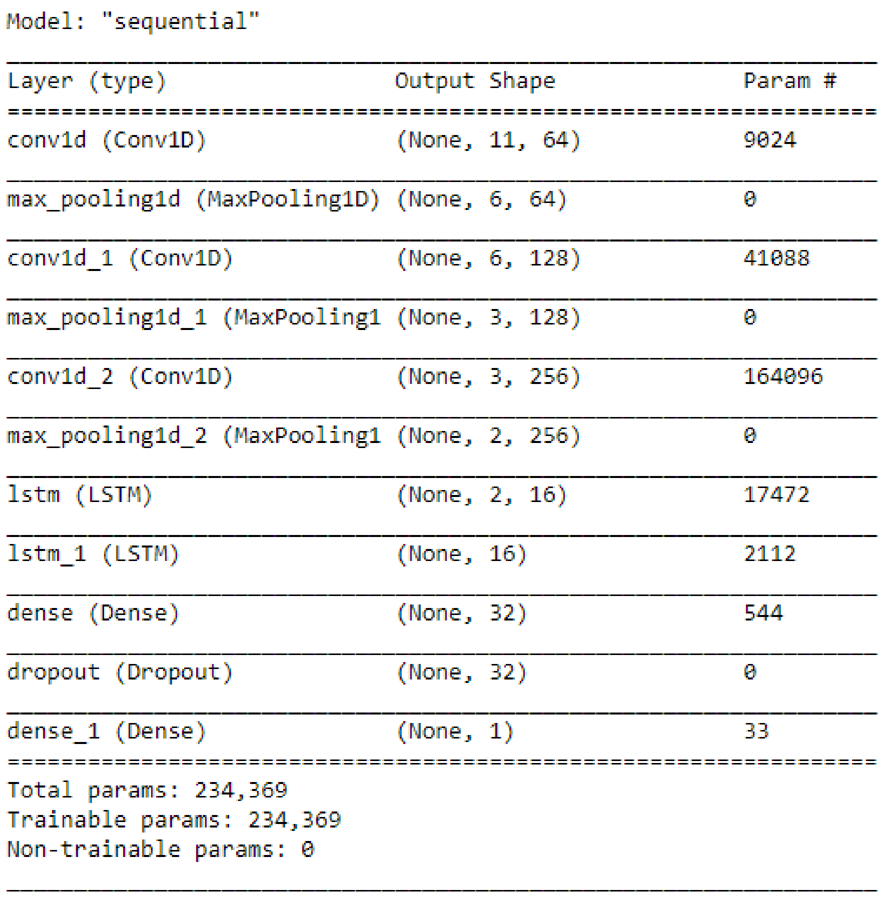


(e) Sequential CNN-LSTM

Figure A.2: Confusion matrices for each model with noise added to data.

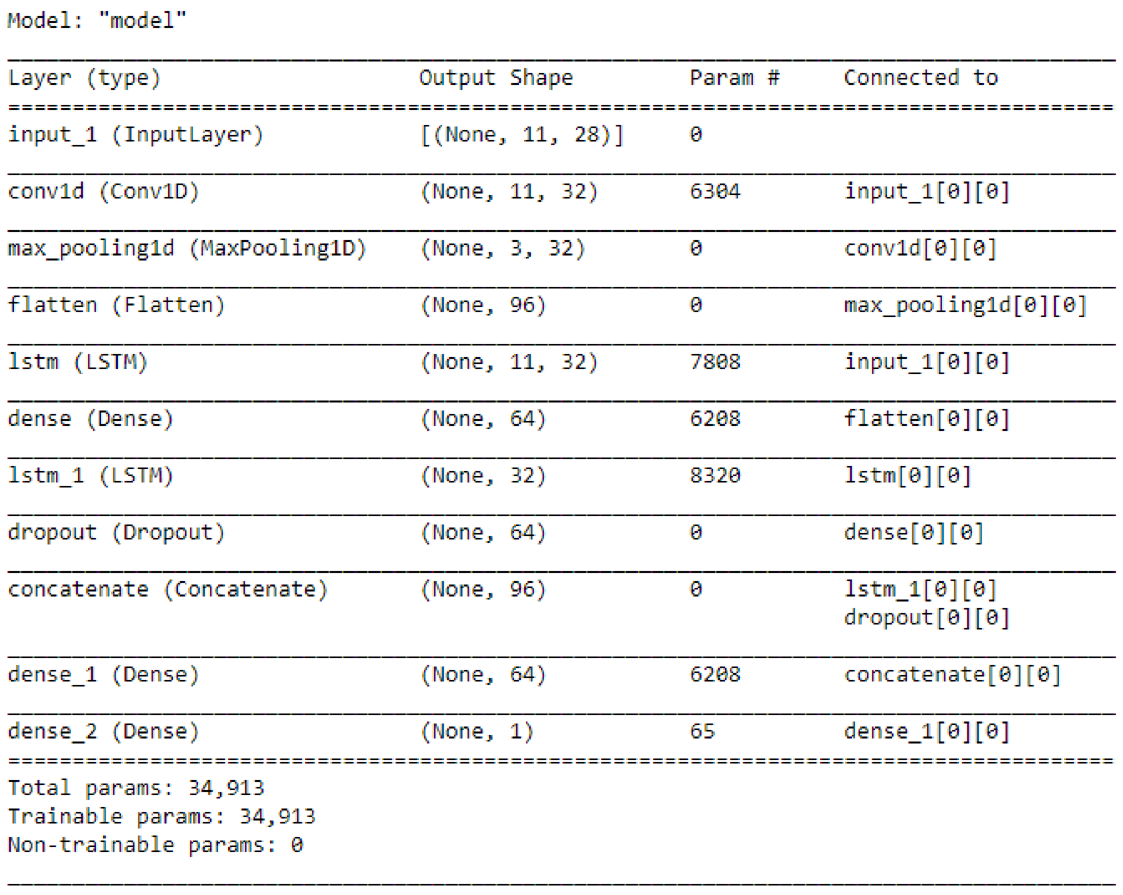
B

Appendix 2



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv1d (Conv1D)              (None, 11, 64)            9024
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 6, 64)             0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 6, 128)            41088
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 3, 128)            0
_________________________________________________________________
conv1d_2 (Conv1D)            (None, 3, 256)            164096
_________________________________________________________________
max_pooling1d_2 (MaxPooling1 (None, 2, 256)            0
_________________________________________________________________
lstm (LSTM)                  (None, 2, 16)             17472
_________________________________________________________________
lstm_1 (LSTM)                (None, 16)                2112
_________________________________________________________________
dense (Dense)                (None, 32)                544
_________________________________________________________________
dropout (Dropout)            (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 33
=================================================================
Total params: 234,369
Trainable params: 234,369
Non-trainable params: 0
_________________________________________________________________
```

Figure B.1: Architecture of the sequential CNN-LSTM.



```
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, 11, 28)]     0

__________________________________________________________________________________________________
conv1d (Conv1D)                 (None, 11, 32)       6304        input_1[0][0]
__________________________________________________________________________________________________
max_pooling1d (MaxPooling1D)    (None, 3, 32)        0           conv1d[0][0]
__________________________________________________________________________________________________
flatten (Flatten)               (None, 96)           0           max_pooling1d[0][0]
__________________________________________________________________________________________________
lstm (LSTM)                     (None, 11, 32)       7808        input_1[0][0]
__________________________________________________________________________________________________
dense (Dense)                   (None, 64)           6208        flatten[0][0]
__________________________________________________________________________________________________
lstm_1 (LSTM)                   (None, 32)           8320        lstm[0][0]
__________________________________________________________________________________________________
dropout (Dropout)               (None, 64)           0           dense[0][0]
__________________________________________________________________________________________________
concatenate (Concatenate)       (None, 96)           0           lstm_1[0][0]
                                                                 dropout[0][0]
__________________________________________________________________________________________________
dense_1 (Dense)                 (None, 64)           6208        concatenate[0][0]
__________________________________________________________________________________________________
dense_2 (Dense)                 (None, 1)            65          dense_1[0][0]
==================================================================================================
Total params: 34,913
Trainable params: 34,913
Non-trainable params: 0
__________________________________________________________________________________________________
```

Figure B.2: Architecture of the parallel CNN-LSTM.