CHALMERS
UNIVERSITY OF TECHNOLOGY

UNIVERSITY OF GOTHENBURG

Driver Behavior Classification in Electric Vehicles

Modeling aggressive driving behavior in electric vehicles with novel deep learning and active learning methods

Master's thesis in Computer science and engineering

FEDERICA COMUNI
CHRISTOPHER MÉSZÁROS

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2021

Master's thesis 2021

Driver Behavior Classification in Electric Vehicles

Modeling aggressive driving behavior in electric vehicles with novel deep learning and active learning methods

FEDERICA COMUNI
CHRISTOPHER MÉSZÁROS

UNIVERSITY OF GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2021

Driver Behavior Classification in Electric Vehicles
Modeling aggressive driving behavior in electric vehicles with novel deep learning and active learning methods
FEDERICA COMUNI
CHRISTOPHER MÉSZÁROS

Supervisor: Niklas Åkerblom, Department of Computer Science and Engineering
Advisor: Niklas Åkerblom, Volvo Cars
Examiner: Morteza Haghir Chehreghani, Department of Computer Science and Engineering

Master's Thesis 2021
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 2021

Driver Behavior Classification in Electric Vehicles
Modeling aggressive driving behavior in electric vehicles with novel deep learning and active learning methods
FEDERICA COMUNI
CHRISTOPHER MÉSZÁROS
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

Studies have shown that driving style affects the energy consumption of electric vehicles, with aggressive driving consuming up to 30% more energy than moderate driving. Therefore, modeling of aggressive driving can provide a more precise estimation of the energy consumption and the remaining range of a vehicle. This study proposes driver behavior classification on vehicle-based measurements through several deep learning models: convolutional neural networks, long short-term memory recurrent neural networks, and self-attention models. The networks have been trained on two naturalistic driving datasets: a labeled dataset generated from a test vehicle on-site at Volvo Cars and unlabeled data collected from co-development Volvo Cars vehicles. The latter dataset has been annotated following rules and driving parameters quantifying the aggressiveness of driving style. The implemented models achieve promising results on both datasets, with the one-dimensional convolutional neural network yielding the highest test accuracy throughout experiments. One of our contributions is to use self-attention and deep convolutional neural networks with joint recurrence plots, which are appropriate for longer sequences because they bypass sequential training. The study also explores several active learning techniques such as uncertainty sampling, query by committee, active deep dropout, gradual pseudo labeling, and active learning for time-series data. These techniques showed variable results, with uncertainty sampling performing consistently better than random sampling. This study confirms the effectiveness of machine learning models in classifying driver behavior. It also shows that active learning can considerably decrease the need for training data.

Keywords: Aggressive driver behavior, Driver behavior classification, Self-attention, Recurrence plots, active learning, Active deep dropout, Gradual pseudo labeling.

Acknowledgements

First of all, we would like to thank our academic supervisor and company advisor Niklas Åkerblom for his continuous support and insightful advice. Thank you for always being available for any question or discussion and for pointing us in the right direction during the implementation of this project. We would also like to thank our examiner Morteza Haghir Chehreghani for his support and guidance and our thesis coordinator Birgit Grohe for her help.

We are also thankful to Volvo Cars for providing this research opportunity. This project would have not been possible without Volvo's support and resources. We also want to extend our thanks to the members of the Volvo Cars team: Ole-Fredrik Dunderberg, Viktor Larsson, Sudhir Rao, David Nigicser, and Sandeep Mewada. Your support and presence have made our experience at Volvo Cars always pleasant and educational.

Finally, we would like to thank our fellow thesis workers at Volvo Cars Avanish Raj and Dhananjay Yadav. Thank you for the interesting discussions and for the fun times throughout these months. We will always be up for a ping pong rematch.

Christopher Mészáros, Federica Comuni, Gothenburg, June 2021

Contents

List of Figures

List of Tables

1

Introduction

Battery electric vehicles (BEVs) present many advantages over internal combustion engine vehicles (ICEVs), including a lighter environmental impact in the majority of vehicle usage scenarios [1]. These advantages have likely contributed to the recent rise of popularity of electric vehicles (EVs), particularly at European level, with the European market share growing by 44% between 2018 and 2019 and with nine European markets ranking in the top ten for penetration rate in 2020 [2]. Despite this success, it is estimated that range anxiety, i.e., the fear that the vehicle will come to a stop due to a depleted battery before reaching the destination or a recharging station, is a strong hindering factor in the transition to EVs in private transportation [3]. This fear is not completely unfounded: while high-end BEV models have an estimated range of over 500 km with a single charge, the real-world autonomy of medium-end models tends to lie within the 200-300 km range (2020) [4].

Range anxiety can be eased by providing the driver with a reliable and accurate estimation of the remaining range: studies have shown that energy consumption is significantly affected by the driver's behavior [5, 6], with moderate driving styles saving up to 30% energy compared to aggressive driving [5]. Driver behavior can be assessed from driving events such as turning and tailgating, and from vehicle-based measures collected by the On Board Diagnostics (OBD) system interfacing with the vehicle's sensors and with the engine control unit (ECU) [7, 8]. Previous research suggests that driving signals, such as vehicle's speed, acceleration and change in acceleration (jerk), can correlate with driving style [6, 8]. Studies also suggest that machine learning models can effectively infer driver behavior from these signals [8, 9].

This study therefore proposes driver behavior classification with four deep learning algorithms: a long short-term memory recurrent neural network (LSTM), a one-dimensional (1D) and a two-dimensional (2D) convolutional neural network (CNN), and a self-attention model. This study also explores active learning and gradual pseudo labeling to optimize the training process and obtain higher accuracy with fewer labeled samples. This optimization is particularly important in the scenario at hand because the manual annotation of time-series data is costly and time consuming. This study is therefore part of Volvo Cars' long-term goal of performing a more accurate estimation of the remaining range in EVs by modeling driver behavior.

1.1 Background

There is a common intuition that there exist different types of driving styles. A driving style that is interesting to analyze is *aggressive driving*. This style consists of a set of hazardous or hasty measures that a driver is willing to take in order to spare time. These measures include tailgating, severe acceleration and deceleration, and risky or sudden lane changes. Other types of driving styles include distracted, drowsy, and drunk driving [10]. Detection of different driver behavior has benefits in several applications (e.g., system notification when the driver appears sleepy). Therefore, modern research is trying to investigate whether *driver behavior* can be modeled using various tools and techniques.

Driving behavior has been studied over the last decade with various models and features. Some modeling methods include *Gaussian Mixture Models* (GMM) [11, 12, 13], *Hidden Markov Models* (HMM) [13, 14, 15], and neural networks [10, 16, 17]. These studies on driver behavior can also differ in the type of data gathered used for modeling. Examples of such measurements are Controller Area Network (CAN) bus signals [13, 17, 18, 19], smartphone sensors [19, 20, 21], questionnaires [22, 23], and video feed [24, 25]. This difference in measurements during data gathering adds ambiguity on what exactly is being modeled. Elamrani Abou Elassad et al. (2020) have introduced a conceptual framework to reduce ambiguity when modeling driver behavior [7] .

Elamrani Abou Elassad et al. (2020) state that there are three modules that affect driver behavior: the *driver*, the *vehicle*, and the *environment*. The driver module can affect the behavior due to their driving profile such as age, driving experience, etc. The vehicle module can affect the behavior due to addition of equipment within a car model (i.e., parking displays, cruise control, etc.). Lastly, the environment module encompasses road condition, weather condition, traffic condition, etc.

Elamrani Abou Elassad et al.'s conceptual framework further breaks down the term *driving behavior* into three types of phenomena: *driving events* (i.e., driving operations performed by the driver such as tailgating, turning, etc.), the *physiological state* of the driver, and the *psychological state* of the driver. These phenomena can be modeled using various metrics mentioned below [7]:

1. **Vehicle-based measures**: these measurements capture the operations of the driver and the vehicle state, e.g. vehicle speed, acceleration, steering wheel, and others. These measurements can be collected via an On-Board Diagnostics module or through a CAN-bus.
2. **Physiological-based measures**: these measurements include putting sensors on the driver to measure heart rate, electrical activity in the brain, and muscle response. [1]

[1] These measurements can be collected with various equipment such as electrocardiogram (used to measure heart rate), electroencephalogram (used to measure brain activity), and electromyogram (used to measure muscle activity).

3. **Behavioral-based measures**: these types of measurements are used to analyze driver's facial expression. This is carried out using a video stream on the driver's face to capture micro expressions while driving.
4. **Subjective measures**: measurements in this category include subjective evaluations. Questionnaires are a common method used to gather subjective insight.

This thesis will mainly focus on modeling *driving events* using *vehicle-based measurements*.

1.2 Related work

Numerous research studies have explored how driver behavior can be identified and how it affects energy or fuel consumption. Several articles have adopted a statistics-based approach to perform modeling of driving features, while others have integrated such statistical analysis with classical or deep machine learning to perform driver behavior classification. These two approaches are covered in the next two subsections. The third subsection covers previous work related to active learning, and the last subsection covers self-attention models.

1.2.1 Statistics-based approaches

Younes et al. (2013) [6] proposed an analysis of factors affecting energy consumption in EVs, including driver behavior. The authors defined a calm driving style as characterized by gradual and gentle accelerations and decelerations. They also defined an aggressive style by stark and heavy accelerations and decelerations, and a normal style by a middle ground between the two, while also cross comparing three route types (city, highway and mountain). The authors stated the importance of controlling for route type when performing driver behavior classification: irregular mountain terrains and winding roads, in fact, might require us to increase the threshold for what is considered aggressive driving. The researchers also computed some parameters that were found to have a strong correlation with driving behavior: positive kinetic energy (PKE), standard deviation of jerk, mean of jerk, relative positive acceleration (RPA), and root mean square of the power factor (RMSPF). These parameters have been found relevant for our study when annotating our data.

The effect of jerk on driving style has also been explored by Feng et al. (2017) [26]. The authors defined aggressive driving as displaying either speeding or tailgating behavior or as being involved in crash or near-crash situations. They found that aggressive drivers tend to display larger jerk standard deviation and that negative jerk in particular is the best predictor of aggressive driving. Constantinescu, Marinoiu and Vladoiu (2010) [27] have also performed analysis on similar driving features, such as mean and standard deviation of both speed and acceleration. The authors also included in their analysis the percentage of time the driver's speed was over 60 km/h, which has been found relevant for our methodology.

There are also several studies that look for correlation between driver behavior and energy or fuel consumption. One such study is the one by Bingham et al. [5], which proposed a quantitative estimation of the effect of driving style on energy consumption, and found that aggressive driving can consume up to 30% more energy than moderate driving. Similarly to [6] and [26], they also found that aggressive drivers show larger standard deviation of the acceleration.

1.2.2 Machine learning-based approaches

One of the earliest applications of neural networks to the identification of driving styles is the one proposed by MacAdam et al. (1998) [16], where aggressiveness was measured through the headway space between and the acceleration towards the vehicle in the front. The researchers categorized driving segments into five levels of aggressiveness. More recently, Júnior et al. (2017) [20] proposed a smartphone sensors-based detection of aggressive driving events, e.g., acceleration, turning, and lane changes. The authors compared several machine learning classifiers and window sliding sizes for data processing. Lin et al. (2019) [28] proposed a comparison of feature statistical distribution maps to identify and evaluate drivers from driving events. Like Júnior et al. [20], they made use of gyroscope and accelerometer data describing acceleration, braking, lane changes and turning.

Shahverdy et al. (2020) [10] classified driving style into five categories: distracted, drowsy, drunk, aggressive, and safe. They employed a novel approach by using 2D convolutional neural networks and by converting driving signals into recurrence plots. The authors argued that the spatial information in the recurrence plots has more benefits than temporal information in driving signals. One of the benefits is efficient training, as these types of networks can bypass sequential training. Sequential training for very long time windows is not feasible. Converting driving signals into recurrence plots is one of the methods investigated in our thesis.

Halim et al. (2016) [29] profiled driver behavior through clustering of driving features and performed classification with a neural network and a support vector machine (SVM). The authors found that average and maximum speed, as well as number of times braking and honking the horn occur, were significant features for profiling drivers.

Several studies applied LSTMs: Ping et al. (2019) [8] combined environmental information from video footage with driving features from the ECU, and classified windows of data into three classes, corresponding to three fuel consumption levels. Xing et al. (2020) [9] performed statistical analysis of driving features and confirmed the previous finding [5] that aggressive driving, characterized by larger mean and standard deviation of the acceleration, yields higher energy consumption. The authors used the same features to perform prediction of vehicle's velocity and energy consumption index, reaching the lowest root mean square error with LSTM-based models.

1.2.3 Active learning

Active learning, i.e., the optimization of a classifier's training procedure by selecting the most informative samples for the training set, has been subject of extensive studies since the 1980s. Examples of its applications include tasks in which annotation might be expensive or scarce, such as text classification [30], speech recognition [31], and cancer diagnosis [32]. In 2001, Tong [33] explored the application of several active learning methodologies to classification with SVMs and parameter estimation and causal discovery with Bayesian networks, demonstrating that active learning can successfully decrease the need for training data in those scenarios.

Conventional active learning methods include uncertainty sampling and query by committee [34]. The former queries the most uncertain samples (according to an informativeness measure) to a human oracle for annotation. The latter queries the most informative samples by using a committee of models. The queried samples are the ones the committee disagrees the most upon.

Extensive research on the topics of uncertainty sampling and query by committee has been conducted in recent years. In 2015, Gammelsæter [35] proposed a technique called active deep dropout (ADD). The idea behind this technique is to implement a committee of models through dropout regularization. More recently, Bossér et al. (2020) [36] performed a comprehensive analysis of active learning methodologies by comparing several techniques and informativeness measures on the CIFAR-10, MNIST and Fashion-MNIST datasets. The authors also developed an algorithm incorporating active and semi-supervised learning, which they named *gradual pseudo labeling algorithm (GPLA)*. Further details about GPLA are covered in subsection 2.3.5.

In 2017, Peng, Luo and Ni [37] proposed a novel approach (named *ACTS*) of active learning to time-series data by defining a new informativeness measure based on the concepts of uncertainty and utility. Further details about this approach are covered in section 2.4.

1.2.4 Self-attention networks

Vaswani et al. (2017) proposed the transformer model for natural language processing (NLP) tasks [38]. The main contribution in their study is the *self-attention* module used in sequence modeling. The benefit of this model is its ability to perform parallel computations in sequence modeling tasks. Earlier architectures for sequence modeling are recurrent neural networks, gated recurrent unit recurrent neural networks (GRU), and long short-term memory neural networks (LSTM). These architectures require sequential training of the inputs, which is not feasible for very long sequences. The authors argue that there is a paradigm shift towards attention-based models (hence the title of the study "*Attention is all you need*") and a paradigm shift away from sequential training.

Mahmud et al. (2020) built upon the ideas from Vaswani et al. by including self-attention blocks in their architectures. Although the original self-attention module was used for NLP tasks, Mahmud et al. [39] argue that words are equivalent to sensor values and that sentences are equivalent to time windows. Hence, they employed self-attention to model Human Activity Recognition (HAR) [39].

1.3 Aim

The project aims to perform driver behavior classification while optimizing the training process with active and semi-supervised learning. It therefore answers the following research questions:

> *Can deep learning architectures achieve good accuracy on the driver behavior classification problem?*
>
> *Can the investigated active learning approaches achieve good accuracy with less labeled data?*

This study proposes several contributions. First, it investigates self-attention networks and 2D CNNs with joint recurrence plots on driver behavior classification. Secondly, it applies novel active learning approaches, such as ADD and GPLA, to this task.

1.4 Limitations

Energy consumption and driver behavior in EVs are often dependent on external factors, such as weather and road condition. A 2016 literature survey of factors affecting fuel consumption and CO_2 emissions found that, while aggressive driving increased fuel use by a median 26%, the presence of rain and heavy traffic increased it by 30% [40]. This project, however, only focuses on information collected by the vehicle's sensors and ECU and does not include any data from the environment around the vehicle. The reason for this limitation is that assessing the impact of each factor on energy consumption requires controlling for all other variables, and therefore excessively broadens the scope of the study for the time at disposal.

Furthermore, studies have shown that video footage of the driver's face can be useful in identifying behavior through eye movements [41]. The project does not make use of it because this type of data is not consistently present across datasets.

Regarding driving behavior, different studies present slightly different definitions of aggressiveness; the general consensus, however, is that aggressive driving is defined by strong accelerations and decelerations, speeding, and a tendency to tailgate. This study takes into consideration all three of these aspects but does not consider occurrence of accidents, as done by Feng et al. [26].

1.5 Ethical considerations

The project entails three main stakeholders: the drivers, Volvo Cars, and the general public. The following section will outline some of the ethical implications for the three parties.

1.5.1 Drivers

A study from 2012 by Cahour et al. analyzed the reactions of drivers borrowing an EV for two weeks from an emotional and cognitive stand point [42]. From the study it emerged that drivers can display a wide variety of behaviors and driving styles as a consequence of the EV's reduced autonomy compared to traditional vehicles: drivers tended to be less adventurous with an EV and to use it only for well-known or planned-ahead journeys. Drivers reported being afraid or anxious of not being able to reach the preset destination, and being more stressed when the controls signaled low battery level. In some cases, drivers adopted drastic behaviors to preserve battery life, such as turning off all unnecessary appliances within the vehicle. A more accurate prediction of the remaining range can therefore have a positive psychological and emotional impact on the most easily stressed drivers. Also, if the project can directly or indirectly help drivers become aware that aggressive driving implies a higher battery consumption, it might nudge them to adopt a calmer (and safer) driving style.

1.5.2 General public

A study by Kester from 2019 reports that range anxiety is an important hindrance to the general public's adoption of EVs, and argues that a more accurate estimation of the remaining range can indirectly favor the popularity of EVs and the purchase of smaller models, thus contributing to environmental sustainability [3]. This is especially relevant, the study reports, because drivers tend to overestimate the length of their journeys.

1.5.3 Volvo Cars

Finally, ethical concerns can arise regarding driver identification and the handling of users' data. One of the datasets used in this study is collected from Volvo Cars employees' leased vehicles and from test vehicles available on-site at Volvo Cars. The employees, upon signing of the lease contract, are informed about the type of data collected and its use. The procedure performs pseudonymization and is compliant with GDPR. Each employee is also required to inform other drivers about the data collection.

1.6 Structure of the thesis

The report is structured as follows: chapter 2 explores the theoretical principles of the algorithms and technologies employed in this study; chapter 3 illustrates

the data collection, generation and processing procedures adopted, as well as the architectures of the neural models; chapter 4 showcases the results of the study and finally, chapter 5 discusses the results and possible future work.

2
Theory

This section provides a theoretical overview of the algorithms employed for this study, as well as of the mathematical foundations behind them. Section 2.1 describes the neural networks adopted in the classification task, section 2.2 outlines the kernel density estimation method employed in the data annotation process, and section 2.3 explains the principles of active learning and related methodologies.

2.1 Supervised learning

In supervised machine learning, the objective is to approximate an unknown function $y = f(\boldsymbol{x})$ mapping each input sample $\boldsymbol{x_i}$ to the corresponding output y_i, for a set of n input-output pairs $(\boldsymbol{x_1}, y_1), ..., (\boldsymbol{x_n}, y_n)$ [43]. The approximation function $\hat{f}$ is in this case computed by a mathematical model called artificial neural network, whose fundamental component, the neuron, is inspired by the biological neuron underlying the nervous system of most animals. Artificial neural networks approximate $f(\boldsymbol{x})$ by defining $y = \hat{f}(\boldsymbol{x}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ represents the learnable parameters of the function. The artificial neuron was first introduced in 1958 by American psychologist Frank Rosenblatt, who named this type of model *perceptron* [44]. Perceptrons allow the representation of linearly separable functions by defining function $f(\boldsymbol{x})$ as described in equation 2.1:

$$f(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{w} \cdot \boldsymbol{x} + b \geq 0, \\ 0 & \text{otherwise} \end{cases} \tag{2.1}$$

where $\boldsymbol{x}$ is the input vector, $\boldsymbol{w}$ is the vector of weights, $\boldsymbol{w} \cdot \boldsymbol{x}$ is the dot product such that $\boldsymbol{w} \cdot \boldsymbol{x} = \sum_{i=1}^{n} w_i x_i$, where n is the number of input elements, and b is the bias, a scalar value used for the shifting of the decision boundary. Weights and biases are the adjustable parameters that we previously defined with $\boldsymbol{\theta}$. The sum of the dot product and the bias is passed through the Heaviside step function, which mathematically represents the neuron "firing" once a set threshold, in this case 0, is reached. The Heaviside function is one of the many *activation functions* adopted in neural networks to set the range of the neurons' output in each layer. This simple model is limited by its inability to represent non linearly separable functions: the function approximated by a perceptron will always be linear [44]. To obviate this problem, artificial neural networks concatenate perceptrons in *layers* where the input to each neuron at any layer is composed of the output of all neurons at the previous layer. This basic form of artificial neural network is typically called *feed-forward*, because it does not involve any feedback mechanism where outputs are re-immitted

as input [43]. This concatenation of layers is therefore a composition of functions, where the final output is dependent on the output from all layers, as represented in equation 2.2 for a network with two layers [43]:

$$\boldsymbol{y} = \hat{f}_2(\hat{f}_1(\boldsymbol{x}; \boldsymbol{\theta}); \boldsymbol{\theta}) \tag{2.2}$$

The presence of *hidden*, i.e. intermediate, layers, as well as the use of non-linear activation functions, allows artificial neural networks to approximate non-linear functions. Figure 2.1 shows the graph of a feed-forward neural network with one hidden layer: biases and weights are shown on the corresponding nodes and edges, respectively.

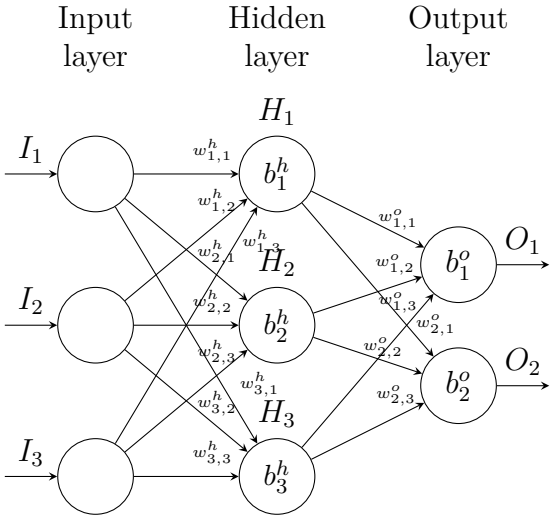


Figure 2.1: A feed-forward neural network with one hidden layer. I_i, H_i and O_i denote the numbered neurons in the input, hidden and output layer respectively, $w_{i,j}^h$ and $w_{i,j}^o$ denote the weights applied to the input from neuron j to neuron i in the hidden and output layer respectively, and b_i^h and b_i^o denote the bias applied in neuron i on the hidden and output layer respectively.

The adjustment of the learnable parameters is performed according to two fundamental components: a cost function and an optimization algorithm. The cost function quantifies the difference between the actual and the predicted values (i.e., the error - see equation 2.3 for the sum of squared errors), while the optimization algorithm tunes $\boldsymbol{\theta}$ to minimize the cost function.

$$E = \sum_{i=1}^{n} (\hat{f}(\boldsymbol{x_i}; \boldsymbol{\theta}) - y_i)^2 \tag{2.3}$$

If the cost function is differentiable, as is usually the case in machine learning, the optimization is gradient based and finds a local or the global minimum of the cost function by computing its partial derivative with respect to each parameter. The iterative update of the parameters is computed in backward direction, from the output to the input layer; for this reason, it is typically called *back-propagation*. Equation 2.4 shows a single gradient descent update of parameters θ_{ij} applied to the input from node j to node i:

$$\theta_{ij} \leftarrow \theta_{ij} - \eta \frac{\partial E}{\partial \theta_{ij}} \tag{2.4}$$

Where η is a scale factor called *learning rate* determining the rate at which the parameters are updated.

2.1.1 Dropout

A neural networks with a lot of parameters can easily *overfit*. The term overfitting describes when the neural network performs well on the training data, but is unable to perform well on new, unseen data. There are several *regularization* techniques that are used to control overfitting.

Dropout is a regularization technique that drops certain neurons during training with probability p. This can be implemented with a binary vector at each layer (except for input and output layer). Backpropagation during dropout works in similar fashion, as mentioned in section 2.1. The difference is that the neurons that have been dropped will be zeroed out during back propagation. However, during the prediction phase, the weights of a layer will be scaled according the the probability of the neuron being dropped.

2.1.2 Convolutional neural networks

Convolutional neural networks are a special type of neural network optimized to process data in a *"grid-like topology"* [43] by employing convolutions instead of the dot product found in regular neural networks. In mathematics, convolution is defined as the integral of a function sliding over another function, i.e., the amount of overlap between the two functions. It is described by equation 2.5 [45]:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{2.5}$$

Where f and g are two functions involved in the convolution. In machine learning, convolution is implemented by "scanning" windows of the input space with units, i.e. vectors or matrices of weights, and by forcing these units to keep the same weight configuration across the whole space [46]. This brings two main advantages: a significant reduction in number of parameters, which would grow prohibitively large if, for example, weights were applied to each single pixel in an image, and the invariance of the network to geometric transformations in the input. The convolution of units, typically called *filters* or *kernels*, on the input produces a *feature map*. Figure 2.2 shows an abstract representation of a convolutional neural network with 2 layers. In the figure, filters can be seen applying convolutions to a subset of the input, called *local receptive field*.

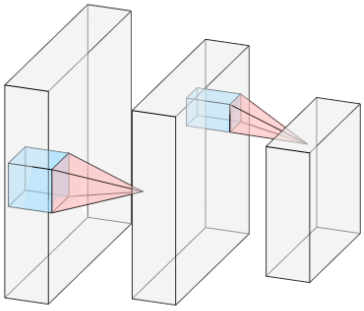


Figure 2.2: The basic structure of a CNN with filters applying convolutions onto a 3D input as it could be, for example, for colored images.

The replication of weights across input space and the focus on local receptive fields allows filters in the early layers of the network to detect low-level patterns, such as corners or edges for images, and filters in late layers of the network to combine these low-level features into higher-order features, such as faces or shapes. Often, convolutional layers are placed before *pooling* layers, which further downsample the feature maps and extract salient features by selecting the highest (max pooling) or the average (average pooling) of the values in the pooling window. Due to their great effectiveness in processing images, convolutional neural networks have become the state-of-the-art for computer vision tasks such as object detection and image segmentation.

2.1.3 Recurrent neural networks

The processing of sequential types of input, as is the case for natural language and time-series data, can benefit from parameter sharing across different time steps. This is because a model with parameters bound to each single time step might be unable to generalize well on sequences of different lengths or with tokens in different positions [43]. Neural networks can share weights by keeping a *hidden* state $\boldsymbol{h}$ for each time step t and using it as input to subsequent steps together with new input $\boldsymbol{x}$, as expressed in equation 2.6 [43]:

$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}) \tag{2.6}$$

At the first time step there is no hidden state from previous steps, hence at any time step $t > 1$ function $\boldsymbol{h}^{(t)}$ can be unfolded as a function $g^{(t)}$ dependent on the input from all previous time steps [43]:

$$\boldsymbol{h}^{(t)} = g^{(t)}(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t-1)}, \dots, \boldsymbol{x}^{(1)}) \tag{2.7}$$

This feedback input of states across time steps defines a recurrence, hence the name *recurrent neural networks* (RNN) for this type of network [43]. Weights are then applied to hidden states and input values for adjustment through back-propagation, as shown in equation 2.8 [43]:

$$\boldsymbol{h}^{(t)} = \tanh(\boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}) \tag{2.8}$$

where $\boldsymbol{W}$ and $\boldsymbol{U}$ are weights applied to the hidden state from the previous time step and to the current input, respectively, and *tanh* is the hyperbolic tangent function, a common activation function in RNNs. Several architectures have been proposed for RNNs; a popular version produces an output at each time step and propagates the hidden states through steps. However, due to the frequent multiplication with shared parameters typical of recurrence, basic forms of RNNs are vulnerable to weights exponentially decreasing as the length of the dependencies, i.e., the distance in time between input values, increases. The decrease in magnitude of the weights significantly reduces the gradient of the cost function w.r.t. the same weights, thus bringing smaller and smaller updates to the parameters at each step and prolonging the training process. To obviate this problem, Hochreiter and Schmidhuber introduced the long short-term memory model (LSTM) in 1997 [47], later improved by Gers et al. [48]. LSTMs replace the regular hidden units of RNNs with *cells*, i.e., operational blocks including internal recurrence of the state and a set of neurons defined as *gating units*. The internal recurrence, also called self-loop, passes through a forget gate determining whether the information should be kept or discarded. Other gates include the external input gate and the output gate, applied to the cell's input and output, respectively, as shown in Figure 3.7. In the figure, H denotes the Hadamard matrix product, $+$ denotes element-wise addition, and *sig* denotes the sigmoid activation function.

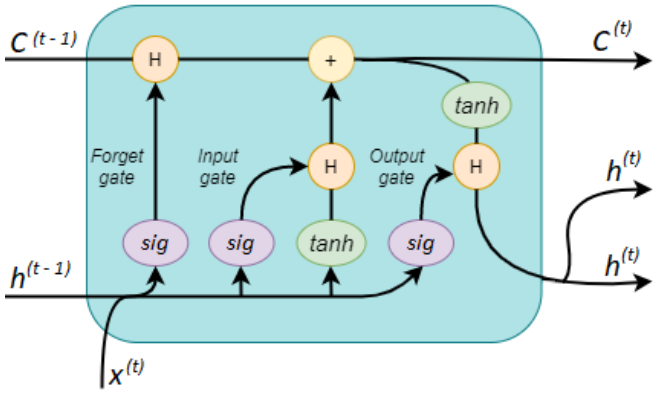


Figure 2.3: An LSTM cell depicting the flow of information and gates.

$C^{(t)}$ denotes the cell's state, given as:

$$C^{(t)} = f^{(t)} \circ C^{(t-1)} + i^{(t)} \circ \tanh(\boldsymbol{W_C} \cdot [\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}] + b_C) \tag{2.9}$$

$f^{(t)}$ denotes the output of the forget gate:

$$f^{(t)} = \sigma(\boldsymbol{W_f} \cdot [\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}] + b_f) \tag{2.10}$$

and $i^{(t)}$ denotes the output of the input gate:

$$i^{(t)} = \sigma(\boldsymbol{W_i} \cdot [\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}] + b_i) \tag{2.11}$$

The flow control in LSTM cells helps counteract the vanishing gradient problem encountered in RNNs; LSTMs are in fact more capable of learning long-term dependencies and have become the most popular version of recurrent network.

2.1.4 Self-attention networks

Attention is a mathematical model employed in neural networks to reproduce the ability of primates to efficiently process sensory input by focusing on certain subsets of it at a time [49]. In self-attention networks proposed by Vaswani et al., each input can interact with itself to compute where attention should be focused. We will first cover simple self-attention before covering the self-attention proposed in [38].

A simple self-attention model can be thought of as the weighted average of the inputs using a weight matrix $\boldsymbol{W}$. Let $\boldsymbol{X}$ be the matrix containing the inputs. The weight matrix $\boldsymbol{W}$ can computed in the following manner:

$$\boldsymbol{W} = softmax(\boldsymbol{X^T} \cdot \boldsymbol{X})$$

This weight matrix can thereafter be used to compute the weighted average of the inputs (noted as $\boldsymbol{Y^T}$) by simply multiplying the weight matrix with the input $\boldsymbol{X}$:

$$\boldsymbol{Y^T} = \boldsymbol{W} \cdot \boldsymbol{X^T}$$

The simple self-attention can be further extended by introducing learnable weights $\boldsymbol{W_Q}, \boldsymbol{W_K}, \boldsymbol{W_V} \in R^{d \times d}$ where d is the dimension of the input. These learnable weights can be multiplied with the inputs to create the following matrices:

$$\begin{aligned} \boldsymbol{Q} &= \boldsymbol{W_Q} \cdot \boldsymbol{X} \\ \boldsymbol{K} &= \boldsymbol{W_K} \cdot \boldsymbol{X} \\ \boldsymbol{V} &= \boldsymbol{W_V} \cdot \boldsymbol{X} \end{aligned} \tag{2.12}$$

Matrices $\boldsymbol{Q}, \boldsymbol{K}$, and $\boldsymbol{V}$ are known as query, keys, and value matrices[1]. The word "self" in self-attention comes from the fact that these matrices are generated from the input itself.

The study by Vaswani et al. introduced the concept of scaled dot-product attention, a particular attention obtained by performing the dot product of each input with three weight matrices. The result is *query* matrix $\boldsymbol{Q}$, *key* matrix $\boldsymbol{K}$ and *value* matrix $\boldsymbol{V}$ as mentioned in Equation 2.12. The product of $\boldsymbol{Q}$ and $\boldsymbol{K}$ is scaled by the square root of the dimension of the key matrix d_k to prevent it from becoming too large and incurring in the vanishing gradient problem, as shown in equation 2.13 [38]:

$$Attention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \underbrace{softmax\left(\frac{\mathbf{Q} \cdot \mathbf{K}^T}{\sqrt{d_k}}\right)}_{\text{weighted average}} \cdot \mathbf{V} \tag{2.13}$$

The authors also improved the model's capability to attend to words at different positions simultaneously by applying multi-head attention, which entails linearly projecting the query, key and value matrices h times and applying scaled dot-product attention to each projection in parallel.

[1] These naming conventions are supposed to represent a soft dictionary where there should be a large similarity between the key and the query.

Positional encoding is introduced to account for positional information when feeding each input through the model. Positional encoding is defined as a function $f : \mathbb{N} \to \mathbb{R}^d$ that converts position of the input into output vector $\vec{p_t}$. This positional vector is later added to the input via simple vector addition.

Let t be the position of the input in a given sequence (the sequence refers to either a time window of sensor values or a sequence of words). Let d be the dimension size of the input and i be the current position of the dimension. Let $j = \lceil \frac{i}{2} \rceil$. Then positional encoding p_{t_i} proposed by [38] can be noted as follows:

$$p_{t_i} = f(t,i) = \begin{cases} \sin(\frac{t}{10000^{2j/d}}), & \text{if } i \text{ is even} \\ \cos(\frac{t}{10000^{2j/d}}), & \text{if } i \text{ is odd} \end{cases} \tag{2.14}$$

The denominator $10000^{2j/d}$ inside each trig function can be noted as $w_j = \frac{1}{10000^{2j/d}}$. The positional embedding $\mathbf{p}_t$ can also be shown in expanded form as follows

$$\mathbf{p}_t = \begin{bmatrix} f(t,1) \\ f(t,2) \\ \\ f(t,3) \\ f(t,4) \\ \\ \vdots \\ \\ f(t,d-1) \\ f(t,d) \end{bmatrix} = \begin{bmatrix} \sin(\frac{t}{\omega_1}) \\ \cos(\frac{t}{\omega_1}) \\ \\ \sin(\frac{t}{\omega_2}) \\ \cos(\frac{t}{\omega_2}) \\ \\ \vdots \\ \\ \sin(\frac{t}{\omega_{d/2}}) \\ \cos(\frac{t}{\omega_{d/2}}) \end{bmatrix} \tag{2.15}$$

Due to the absence of recurrent layers, transformers can fully benefit from parallelization. They have also been able to capture long-term dependencies more effectively than RNNs [39]. For this reason, they have recently gained enormous success in natural language processing and time-series applications [39].

2.1.5 HAR model

Mahmud et al. (2020) proposed that the self-attention model can also be applied to sensor values [39]. The original transformer article (mentioned in the previous section) was originally applied to natural language processing tasks. However, Mahmud et al. argue that words are equivalent to sensor values and that sentences are equivalent to time windows. Therefore, they argue that the self-attention module can also be applied to sensor values as done in applications such as Human Activity Recognition (HAR).

The full model can be viewed in Figure 2.4. It can be seen from the figure that Mahmud et al. incorporate ideas from [38], but also extend their model with additional layers such as *sensor attention* and *global attention*. The role of the sensor attention layer is to extract which of the sensors are relevant for the activity. For example, a sensor located around the upper body is more relevant than a sensor located on the

lower body for standing activities such as talking and eating. The sensor attention layer consists of first transforming the time signals in a window into a single channel image, thereafter applying k amount of 2D convolutional filters. The resulting image is re-converted to a single-channel image by applying 1×1 convolutional filters. This outputs an attention score for each sensor that indicates the relevance for a given activity.

The function of the global attention layer is to compute weighted averages for all time steps in a time window. This computation is performed by using simple self-attention covered in subsection 2.1.4.

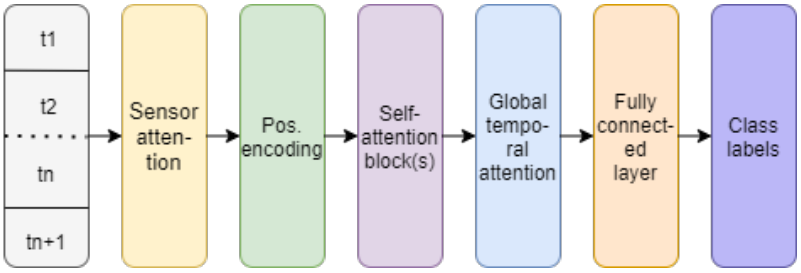


Figure 2.4: HAR model from [39].

2.2 Kernel density estimation

Kernel density estimation is a non-parametric technique employed to estimate the probability density function of a variable from a set of samples. Given a set $\{m_1, m_2, ..., m_n\}$ of n observations sampled from a univariate distribution with unknown density p, the kernel density estimation technique applies a smooth, symmetric function K to each sample, and computes the average of the results for all samples. For bivariate data (a simplification of the multivariate data employed in this study), each sample can be described by its coordinates $\{x_i, y_i\}$ where $i = \{1, ..., n\}$. Radial kernel estimation, a common version of kernel density estimation for multivariate data, computes the probability density function $\hat{p}_n(x, y)$ through the Euclidean distance between any point $\{x, y\}$ and sample point $\{x_i, y_i\}$ as described in equation 2.16 [50]:

$$\hat{p}_n(x,y) = \frac{1}{nh_xh_y}\sum_{i=1}^{n} K\left(\sqrt{\left(\frac{x_i - x}{h_x}\right)^2 + \left(\frac{y_i - y}{h_y}\right)^2}\right) \tag{2.16}$$

where h_x and h_y are two smoothing coefficients, called *smoothing bandwidth*, determining the smoothness of the probability density function, and K might be, for example, the Gaussian function, $K(x) = \frac{1}{\sqrt{2\pi}}e^{-x^2/2}$.

2.3 Active learning

Active learning is a specific approach in machine learning where a model can achieve better performance with a lower amount of training instances (see Figure 2.5). The

instances are queried in a clever way out of a large pool of unlabeled instances (also known as the *pool-based* framework for active learning). The queried instances are then annotated by an oracle (for example, a human annotator). Once these queried instances have been annotated, they can be used to improve the initial model. There are different approaches for querying instances to the oracle; they will be covered in the following subsections.

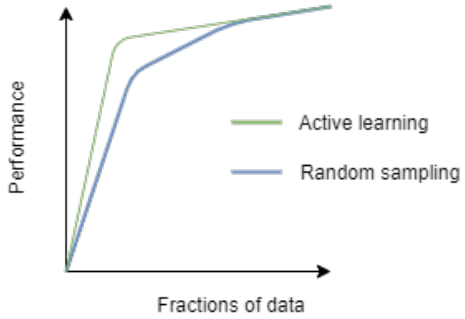


Figure 2.5: Example of a learning curve comparing test accuracy of the same classifier trained with two sampling methods.

2.3.1 Uncertainty sampling

Uncertainty sampling is an active learning approach in which the most "uncertain" instances are queried to the oracle [34]. The "uncertainty" of each sample is computed from an initial machine learning model using some informativeness measure (see subsection 2.3.3). Figure 2.6 compares a classifier trained on actively queried and randomly queried instances on a toy example.

2.3.2 Query by committee

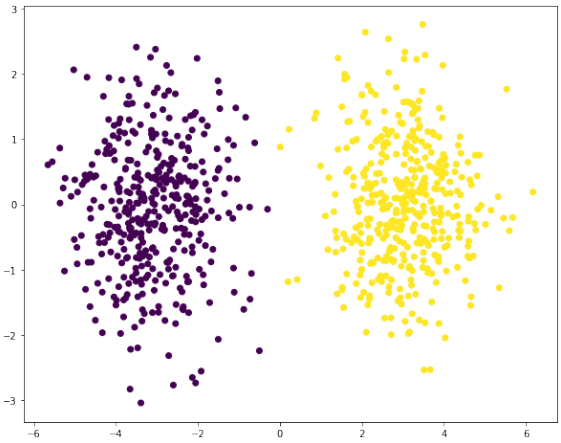
The *query by committee* method queries instances with the help of several models (a *committee* of models). The different models are all trained on the available labeled dataset to make an initial model. However, each model has a different hypothesis/decision boundary which is consistent with the labeled data. The instances on which the committee disagrees the most will be queried to the oracle.

2.3.3 Informativeness measures

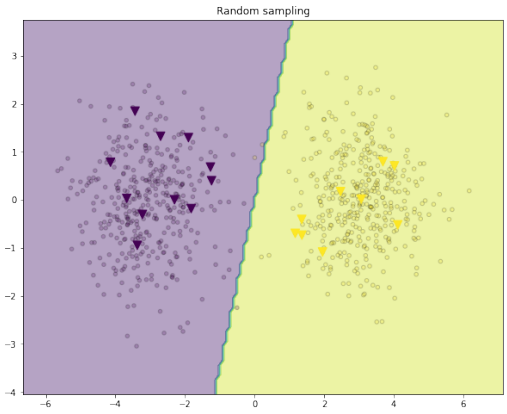
An *informativeness measure* is a heuristic measure used to determine which instances should be annotated from a large pool of unlabeled data. The three most popular informativeness measures are *least confidence*, *margin*, and *entropy* [34].
Least confidence is the simplest informativeness measure and involves choosing instance $\boldsymbol{x}$ in which the model is least confident. The queried instance can be formulated as follows:

$$\arg\max_{\boldsymbol{x}} 1 - P_\theta(\hat{y}|\boldsymbol{x}) \tag{2.17}$$

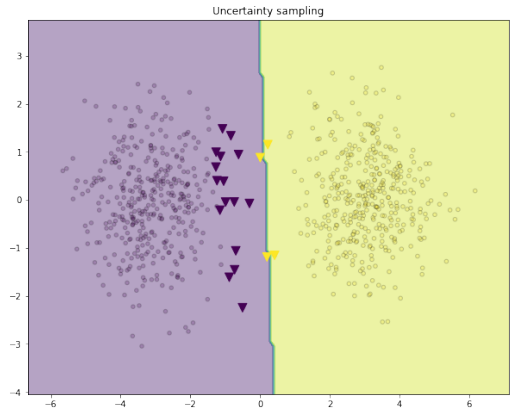
Where $\hat{y} = \arg\max_y P_\theta(y|\boldsymbol{x})$ represents the predicted class label for input $\boldsymbol{x}$ according to the highest posterior probability for the model θ [34]. This informativeness



(a) Scatter plot of two Gaussians



(b) Decision boundary with random sampling



(c) Decision boundary with uncertainty sampling

Figure 2.6: Figure (a) shows a scatter plot of data evenly sampled from two Gaussian distributions. Figure (b) shows a logistic regression classifier trained on 30 randomly drawn samples from (a) (the training instances are noted as upward triangles). Figure (c) shows the same model, trained on 30 samples drawn using uncertainty sampling.

measure only takes into account the class in which the model is least confident, and therefore does not distinguish well between classes in multi-class classification. The following measure takes into account the top two classes:

$$\arg\min_{\boldsymbol{x}} P_\theta(\hat{y}_1|\boldsymbol{x}) - P_\theta(\hat{y}_2|\boldsymbol{x}) \tag{2.18}$$

Where $\hat{y}_1$ is the most probable class and $\hat{y}_2$ is the second most probable class under a model. This informativeness measure is known as *margin* and is better suited than least confident for distinguishing classes for multi-class problems. However, for problems where there is a large amount of class labels, then the margin measure essentially throws away information about the rest of the classes labels. The following informativeness measure takes into account all class labels:

$$\arg\max_{\boldsymbol{x}} -\sum_i P_\theta(y_i|\boldsymbol{x}) \log P_\theta(y_i|\boldsymbol{x}) \tag{2.19}$$

The above expression is commonly referred to as *entropy* and it is the most popular informativeness measure used within active learning [34]. Informally it can be described as choosing the instance $\boldsymbol{x}$ which has the lowest predictability amongst all classes (and equivalently the highest entropy).

In query by committee, the level of disagreement between committee members is instead typically assessed through the *vote entropy* or the *Kullback-Leibler divergence* informativeness measure [34]. The former is a variation of the entropy informativeness measure previously seen, taking into account the number V of committee members *voting* for each labeling y_i (i.e., predicting y_i as the most probable label) over the size of the committee N, as shown in equation 2.20 [34]:

$$\arg\max_{\boldsymbol{x}} -\sum_{i} \frac{V(y_i)}{N} \log \frac{V(y_i)}{N} \tag{2.20}$$

The latter employs the Kullback-Leibler divergence and therefore considers most informative the queries for which the members' label distribution differs the most from the consensus, as described in equation 2.21, where $\theta^{(n)}$ represents a single member of the committee, and C represents the whole committee [34]:

$$\arg\max_{\boldsymbol{x}} \frac{1}{N} \sum_{n=1}^{N} D(P_{\theta^{(n)}} || P_C) \tag{2.21}$$

where the Kullback-Leibler divergence is defined as:

$$D(P_{\theta^{(n)}} || P_C) = \sum_{i} P_{\theta^{(n)}}(y_i|\boldsymbol{x}) \log \frac{P_{\theta^{(n)}}(y_i|\boldsymbol{x})}{P_C(y_i|\boldsymbol{x})} \tag{2.22}$$

and the consensus probability of y_i being the correct label is defined as:

$$P_C(y_i|\boldsymbol{x}) = \frac{1}{N} \sum_{n=1}^{N} P_{\theta^{(n)}}(y_i|\boldsymbol{x}) \tag{2.23}$$

2.3.4 Active deep dropout

Active deep dropout is a type of *query by committee*. Gammelsæter proposed a way to create several models in a committee by using the dropout regularization technique [35]. The idea is to Monte Carlo sample different smaller networks by applying dropout to the same fully trained network. The amount of times one applies dropout equals the amount of models in the committee. Figure 2.7 shows a visual example of query by committee through active deep dropout.

2.3.5 Gradual pseudo labeling

A model trained on more labeled instances becomes more confident in its predictions. Bossér et al. therefore proposed that the model's predictions be used in training the model once enough data has been annotated [36]. This method suggests generating so-called *pseudo labels* for each training instance in which the model is highly confident. These instances with pseudo labels can then be used to train the model in

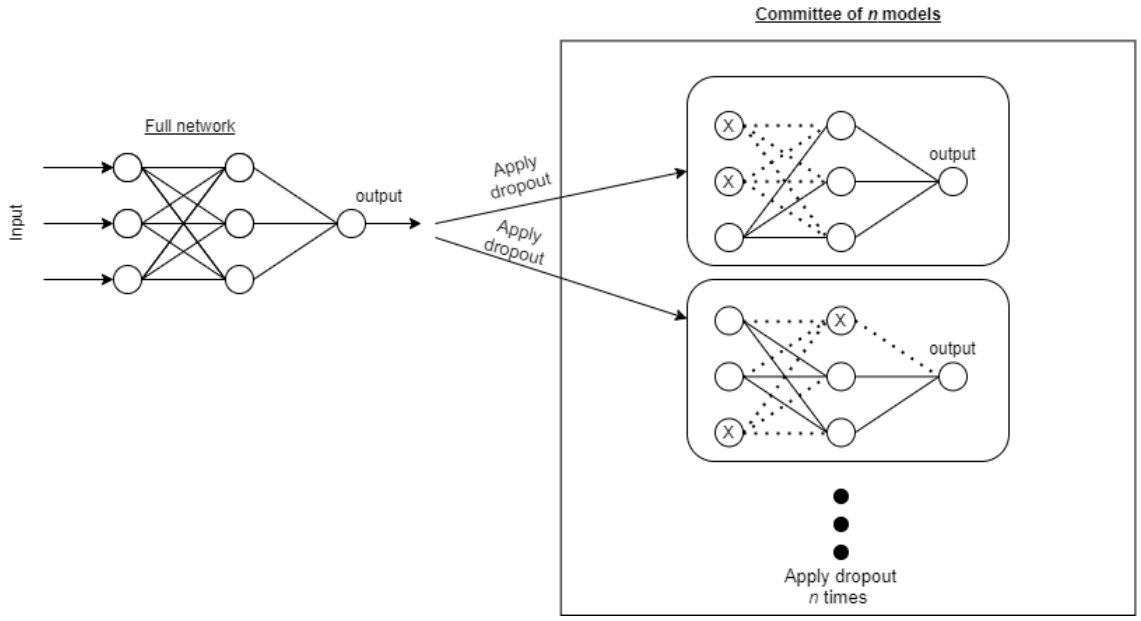


Figure 2.7: Active deep dropout example. The neurons dropped through dropout regularization are noted as ⊗.

a semi-supervised manner. In particular, the pre-trained classifier ranks the pool of unlabeled samples $\boldsymbol{U}$ by the informativeness measure I_i, and then selects the *confident subset* $S_\xi = \{i \in \boldsymbol{U} | I_i < \xi\}$ of unlabeled samples with informativeness measure I_i below threshold ξ. The confident samples are then assigned a pseudo label equal to the class for which the classifier has predicted the highest probability.

2.4 Active learning for time-series data

Peng et al. [37] propose a new informativeness measure which appears to be more suited to active learning on time-series data than traditional measures. The proposed measure consists of *uncertainty* and *utility*. *Uncertainty* combines Shannon entropy with the distance d_i between the sample and its i^{th} closest neighbor, as shown in equation 2.24 [37]:

$$Uncr(\boldsymbol{x}) = -\sum_i P_\theta(y_i|\boldsymbol{x}) \cdot \log(P_\theta(y_i|\boldsymbol{x})) \cdot \frac{d_1}{d_k} \tag{2.24}$$

Utility takes into account the average similarity $\mathrm{Sim}(\boldsymbol{x}, \boldsymbol{y})$ between instance $\boldsymbol{x}$, sampled from subset S of the unlabeled data $\boldsymbol{U}$, and each instance $\boldsymbol{y}$ from the set of $\boldsymbol{x}$'s *reverse neighbors* RN_n (i.e., the samples that have $\boldsymbol{x}$ as one of their n-nearest neighbors), as shown in equation 2.25. This measure has the purpose of querying only instances with a high degree of similarity with their neighbors and to avoid querying outliers.

$$Uti(\boldsymbol{x}|S, \boldsymbol{U}) = 1 + \sum_{\boldsymbol{y} \in RN_n(\boldsymbol{x}) - S} \mathrm{Sim}(\boldsymbol{x}, \boldsymbol{y})/n \tag{2.25}$$

The similarity can be computed through a statistical distance function, such as Jensen-Shannon divergence. The two measures can then be combined to form the novel informativeness measure $I_{\boldsymbol{x}}$ [37]:

$$I_{\boldsymbol{x}} = Uncr(\boldsymbol{x}) + Uti(\boldsymbol{x}|S, \boldsymbol{U}) \tag{2.26}$$

2.5 Signal processing

2.5.1 Sliding time-windows

This technique refers to splitting a larger signal into smaller *time windows*. The idea for this is to extract information from each time window rather than the whole signal itself. The time windows can be used as data for time-series modeling.

There are different types of windowing operations. This section will only focus on *sliding windows* technique. There are two parameters for this technique: *window size* and *step size*. These parameters are explained and visualized in Figure 2.8. A byproduct of these parameters is *window overlap* which is also explained in the figure.

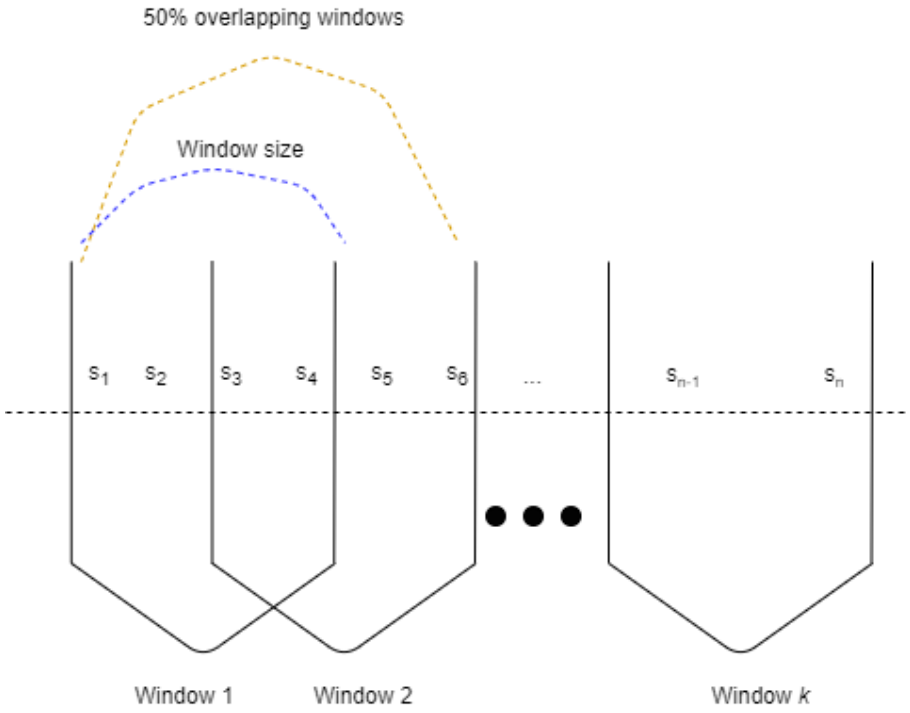


Figure 2.8: Let **s** be a signal with up to n sensor readings at different time steps. Let s_i be a signal reading at time i. The figure above shows signal **s** can be divided into k *time windows*. Window 1 contain s_1, s_2, s_3, and s_4. Window 2 contains s_3, s_4, s_5, and s_6.

2.5.2 Recurrence plots

Signals can be processed into a *recurrence plot*. These plots can visually show if a signal at time i is similar to the signal at time j. A recurrence plot is noted as a matrix where each element can be computed as follows:

$$R(i,j) = \begin{cases} 1, & \text{if } ||\boldsymbol{x}_i - \boldsymbol{x}_j|| \leq \epsilon \\ 0, & \text{otherwise} \end{cases} \tag{2.27}$$

Where $\boldsymbol{x}$ is a signal and ϵ is a threshold that controls the amount of difference between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$.

Joint recurrence plots are a method of combining several recurrence plots into a single one using the matrix operation known as *Hadamard product*. The operation can be noted as $\boldsymbol{A} \odot \boldsymbol{B} = \boldsymbol{C}$. The result is an element-wise multiplication resulting in $\boldsymbol{C}_{ij} = (\boldsymbol{A})_{ij}(\boldsymbol{B})_{ij}$. An expanded version is also shown in Equation 2.28:

$$\underbrace{\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}}_{\boldsymbol{A}} \odot \underbrace{\begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}}_{\boldsymbol{B}} = \underbrace{\begin{bmatrix} a_{11}\,b_{11} & a_{12}\,b_{12} & a_{13}\,b_{13} \\ a_{21}\,b_{21} & a_{22}\,b_{22} & a_{23}\,b_{23} \\ a_{31}\,b_{31} & a_{32}\,b_{32} & a_{33}\,b_{33} \end{bmatrix}}_{\boldsymbol{C}} \tag{2.28}$$

2.6 Evaluation metrics

This study presents a multi-class classification task, therefore the performance of the algorithms is assessed through typical classification evaluation metrics described in the sections below. The learning curve method is employed to compare the performance of the classifiers trained through random sampling and active learning.

2.6.1 Accuracy

The accuracy is the percentage of samples classified correctly, i.e., in a binary classification task, the number of true positives and true negatives over the number of true positives, false negatives, true negatives and false positives, as shown in 2.29 [51]:

$$Acc_b = \frac{TP + TN}{TP + FN + TN + FP} \tag{2.29}$$

For a multi-class classification task, the total accuracy can be computed as the average of the accuracy for each class C_i, as shown in 2.30 [51]:

$$Acc_m = \frac{1}{C} \sum_{i=1}^{C} \frac{TP_i + TN_i}{TP_i + FN_i + TN_i + FP_i} \tag{2.30}$$

In our case, however, it is computed as the frequency with which a model's highest predicted class matches the true class [52].

2.6.2 Confusion matrix

Confusion matrices allow the visualization of the accuracy for each class as a table of the number of predicted labels for each label in the test set [51]. Table 2.1 shows an example of a confusion matrix for 3-class classification.

Actual \ Predicted	**Class A**	**Class B**	**Class C**
Class A	90	4	2
Class B	2	92	5
Class C	6	3	88

Table 2.1: Example of a confusion matrix for the results of a 3-class classification.

2.6.3 Area under the receiver operating characteristic curve

The receiver operating characteristic (ROC) is a graphical plot displaying the ability of a binary classifier to classify correctly for different discrimination thresholds. In particular, the ROC compares the true positive rate and the false positive rate for each given threshold. The true positive rate, also called *recall* or *sensitivity*, is a measure of the classifier's effectiveness at classifying the relevant (or positive) data samples, and is computed as the number of true positives over the number of true positives and false negatives [53]:

$$TPR = \frac{TP}{TP + FN} \tag{2.31}$$

The false positive rate, instead, measures the classifier's tendency to misclassify negative samples, and it is computed as the number of false positives over the number of false positives and true negatives:

$$FPR = \frac{FP}{FP + TN} \tag{2.32}$$

Once the ROC has been estimated, the area under its curve (AUC) can be computed as [54]:

$$AUC = \int_0^1 TPR\, d(FPR) \tag{2.33}$$

and it is an indicator of how likely the classifier is to output a higher value for a randomly picked positive instance than for a randomly picked negative instance, if the output is interpreted as the probability of the sample belonging to the positive class. This estimate can help compare the performance of different classifiers, as shown in Figure 2.9: the classifier with an AUC of 0.847 can be considered less effective than the model with an AUC of 0.934, since, barring the highest false positive rate values, it yields a lower true positive rate at any given threshold.

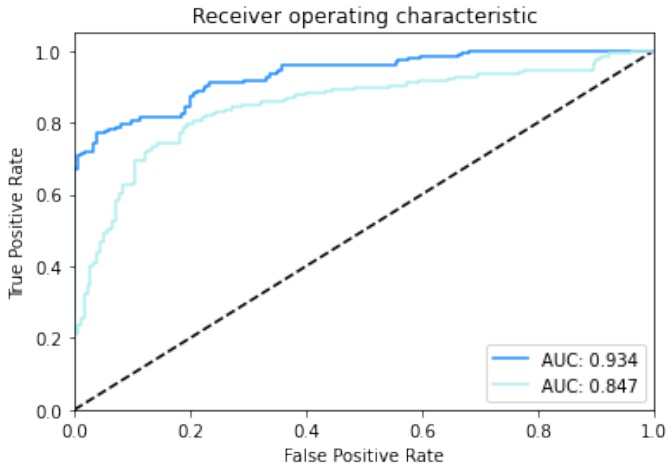


Figure 2.9: ROC and AUC for two classifiers.

2.6.4 Precision

Precision is indicative of how many of the samples that were predicted positive have been correctly classified, and it is computed as the number of true positives over the number of true and false positives [51]:

$$Precision = \frac{TP}{TP + FP} \tag{2.34}$$

For multi-class scenarios, precision is typically computed through *micro-averaging*, i.e. [51]:

$$Precision_{micro} = \frac{\sum_{i=1}^{C} TP_i}{\sum_{i=1}^{C} (TP_i + FP_i)} \tag{2.35}$$

or *macro-averaging*:

$$Precision_{macro} = \frac{\sum_{i=1}^{C} \frac{TP_i}{TP_i + FP_i}}{C} \tag{2.36}$$

The same distinction between micro- and macro-averaging applies to the previously introduced recall, or true positive rate. In case of imbalanced datasets these metrics might instead be averaged by weight, i.e., taking into account the number of true class instances, also called *support*.

2.6.5 F_1 score

The F_1 score is the harmonic average of precision and recall and it is indicative of the model performance when taking into consideration both precision and recall in the same proportion [51]:

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall} \tag{2.37}$$

A micro-averaged F_1 score is the mean of the micro-averaged precision and recall, while the macro-averaged F_1 score is the mean of the macro-averaged precision and recall.

2.6.6 Learning curves

As mentioned in section 2.3, the effectiveness of an active learning strategy, compared to randomly sampling the training set, can be assessed through learning curves. Learning curves, in this case, report the accuracy of the model on the test set for different percentages of training data. A query selection method consistently showing a higher test accuracy for the same percentage of training data is considered more effective than other selection methods.

3
Methods

This section will explain the methodology carried out in the thesis. It can be divided into two main parts: data gathering and processing, covered in section 3.1, and model building, covered in section 3.2.

3.1 Data gathering and processing

This study focuses on the analysis of two naturalistic driving datasets: a labeled dataset generated from a test vehicle on-site at Volvo Cars (see subsection 3.1.1), and unlabeled data collected from co-development Volvo Cars vehicles (see subsection 3.1.2).

3.1.1 Data generation on the test track

Two drivers were asked to drive around the Volvo Cars test track in Torslanda (Gothenburg), on the same BEV model. Each driver could perform a few warm-up laps around the track in order to get familiar with it before any data were collected. An aerial view of the test track can be seen in Figure 3.1: the track is approximately 4.7 km long, with double lanes in each direction. After the warm-up laps, one driver emulated an aggressive and normal driving style, while the other emulated an aggressive, normal and cautious driving style, for an approximately equal number of laps per style. The same style was maintained throughout the whole lap. Each driver was explained how to emulate the driving styles qualitatively, while referencing previous literature and after consulting experts at Volvo Cars. The aggressive driving style was emulated by performing heavy and fast accelerations and decelerations, by changing lane often and quite abruptly, and by driving close to the speed limit. The normal, or less aggressive, style was emulated by performing smoother accelerations and decelerations, by changing lanes only if necessary and gradually, and by keeping a speed slightly below the speed limit. The cautious driving style was emulated similarly to the normal style, but with particularly smooth accelerations and decelerations, and a driving speed significantly lower than the speed limit.

The Vector *CANalyzer* software was used to collect the driving signals from the CAN-bus. The frequency of the sampling was set to 10 Hz. This frequency is well above the average driver response time (which is 2.3 seconds) [56, 57], and as such is able to represent the driver's behavior in detail.

Figure 3.1: Aerial view of Volvo Cars' test track, from Google Earth [55].
Maps Data: Google, ©2021 Aerodata International Surveys, CNES / Airbus, Landsat / Copernicus, Lantmäteriet / Metria, Maxar Technologies.

3.1.2 Data gathering - WICE

WICE (Wireless Information Collection Environment) is Volvo Cars' system for collecting data in test vehicles. The collected data consist of a combination of the car sensors' and navigation map data, and includes information such as the vehicle's longitudinal and lateral speed and acceleration, the current and next speed limit, the road gradient, etc.

Since it is important to control for route type when modeling driver behavior [6], a few pre-processing steps were carried out to isolate a frequent commuting route in the proximity of Volvo Cars (i.e., Hisingsleden). The commuting route can be distinguished almost uniquely in Torslanda by a sequence of speed limit signs on its path. To know the exact distance between speed limit signs, a control drive was performed on the route, and the data from the drive were collected. The distance between signs was found as a product of the average vehicle speed and time elapsed between signs. The distance was then computed for data samples in the WICE dataset and, together with the average road gradient, compared to control data to isolate the desired path.

The data samples used in this study cover a time period of 11 weeks, spanning from

January to April 2021, and are all collected from BEVs. Due to pseudonymization, it is unknown how many drivers have generated the data. Since the samples are not automatically annotated with driving style, an annotation procedure was performed, as described in the following section.

3.1.3 Data annotation

Upon inspection of the collected data, it appeared that several drivers tended to display different behaviors and adopt different driving styles throughout their trips. For this reason, we decided to segment the driving data into sliding windows of 10 or 5 seconds and to perform the driver classification task separately for each window. Figure 3.2 shows one example of inconsistent driver behavior for a single driver on Hisingsleden: from approximately 3 to 5.5 km into the trip, the driver performs relatively smooth accelerations and keeps almost a constant speed, and thus their style might be classified as normal. From there until the eighth kilometer, however, the driver displays more abrupt acceleration and higher speed, thus behaving more aggressively.

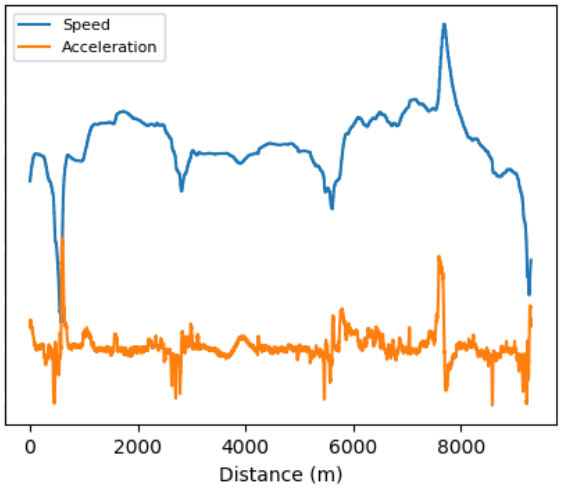


Figure 3.2: Example of driving data from a single driver displaying possible different driving styles.

Several previous studies have performed sliding window segmentation with a certain percentage of overlap between windows [10, 17]. However, overlapping requires a non-random split of the windows into training and test set to prevent data leakage across the sets. Since our dataset presented a non-uniform distribution of classes across drivers and across weeks of data, we opted to use random splitting of non-overlapping windows for the majority of the experiments. The windows were therefore randomly selected across weeks of data while keeping a similar proportion of the three classes, and subsequently randomly split into the training, validation, and test set. Later, in order to assess the performance of the models on overlapping windows, we extracted 5-second windows with 50% overlap for all trips, and selected 20% of the trips presenting a class distribution similar to that of the whole dataset. We designated those trips as test set, while keeping the rest of the dataset as training

set. The windows were annotated using a majority-class system of driving parameters and rules found in accordance with experts at Volvo Cars and in previous work. Table 3.1 shows the set of rules and corresponding class adopted in the annotation process.

Rule	Class
Speeding: driving at least 5 km/h above the speed limit, for at least 20% of the time	If true and slow driving is false: aggressive If true and slow driving is true: normal
Slow driving: driving at least 5 km/h below the speed limit, for at least 10% of the time, when the vehicle in front is at least 20 meters away	If true and speeding is false: cautious If true and speeding is true: normal
Low gap time: when the gap time from the vehicle in the front is 1 second or less, for at least 20% of the time	If true and high gap time is false: aggressive If true and high gap time is true: normal
High gap time: when the gap time from the vehicle in the front is 2.5 seconds or more, for at least 10% of the time, whenever there is a vehicle 50 meters or less in front of the car	If true and low gap time is false: cautious If true and low gap time is true: normal

Table 3.1: Set of rules for annotating the driver behavior based on speed and gap time.

Other parameters considered for the annotation have been previously proposed by Younes et al. [6], and have been found by the authors to correlate with driver behavior: positive kinetic energy (PKE), relative positive acceleration (RPA), root mean square of the power factor (RMSPF), and mean and standard deviation of the jerk. The PKE is a measure of the intensity of positive acceleration manoeuvres, as described by equation 3.1 [6]:

$$PKE = \frac{\sum_i (v_{i+1}^2 - v_i^2)}{D}, v_{i+1} > v_i \tag{3.1}$$

where v_i is the vehicle's speed at timestep i and D is the total trip distance (in our case, the total window distance). The RPA is also described by speed and positive acceleration, as shown in equation 3.2:

$$RPA = \frac{\sum_i (v_i * a_i^+)}{D} \tag{3.2}$$

where a_i^+ is the vehicle's positive acceleration at timestep i. The RMSPF is described as:

$$RMSPF = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (2 * v_i * a_i)^2} \tag{3.3}$$

where $2 * v * a$ is the power factor. These five parameters (PKE, RPA, RMSPF, jerk mean and jerk standard deviation) were computed for windows of 10 seconds of the data generated on the test track, and their Pearson coefficient was calculated to verify if they presented any correlation with driving style. They all presented strong correlation, especially RMSPF and jerk mean. Their Pearson coefficient and two-tailed p-value for testing non-correlation are listed in Table 3.2. The correlation

Parameter	**Pearson coefficient**	**p-value**
PKE	0.246	5.99e-08
RPA	0.21	3.97e-06
RMSPF	0.47	2.53e-27
Jerk μ	0.342	2.03e-14
Jerk σ	0.328	2.69e-13

Table 3.2: Pearson coefficients and p-values for the correlation between driving style and the listed parameters.

is significant because, for the tested sample size of 472 and 470 degrees of freedom, a coefficient of ≈ 0.151 or higher indicates that there is less than 0.1% probability of encountering this level of correlation due to random chance.
In a second phase, the probability density function for Younes et al.'s paramters was estimated through Gaussian kernel density estimation, from the samples computed on the windows of test track data. Subsequently, the same parameters were computed for each window extracted from the unlabeled WICE data, and the probability of the feature value belonging to each class was inferred from the density function. The class with the highest probability for that value was treated as a label. Finally, each window was annotated as belonging to the class that was most represented among the rules shown in Table 3.1 and the probability of the five driving parameters proposed by Younes et al.

3.1.4 Splitting of the data

Table 3.3 summarizes the number of windows per class for each dataset obtained from the window segmentation process. It can be noted that the number of 5-second, non-overlapping windows is less than double the number of 10-second, non-overlapping windows. This is because the annotation system labeled fewer 5-second windows as aggressive, and this was taken into consideration when randomly selecting the windows for the dataset. Stratified random splitting of the non-overlapping datasets into $60 : 20 : 20$ and $80 : 10 : 10$ ratios between the training, validation and test set showed some variance in the models' performance throughout experiments. Therefore, we performed stratified 5-fold cross validation in order to account for this variance, and set the same random seed across experiments. Correspondingly, we defined the same set of seeds for experiments with active learning.

Dataset	Nr. of aggressive windows	Nr. of normal windows	Nr. of cautious windows	TOTAL
WICE 10-second windows	918	1247	1247	**3412**
WICE 5-second non-overlapping windows	1540	1775	1775	**5090**
WICE 5-second windows with 50% overlap	3057	3311	3567	**9935**
Test track	155	123	133	**411**

Table 3.3: Class distribution in the datasets.

3.1.5 Driving features for the neural models

When training the neural models, we selected the driving features that depend the most on the driver's actions (with the exception of speed limit), for a total of 8 features: longitudinal acceleration, speed, speed limit, percentage of pressure on the acceleration pedal, lateral acceleration, steering wheel angle, speed of the steering wheel, and distance from the vehicle in the front. The features were scaled through standardization, i.e., the feature-wise mean μ was subtracted from each single data sample x_i, and the result was divided by the feature-wise standard deviation σ:

$$x_i \leftarrow \frac{x_i - \mu}{\sigma} \tag{3.4}$$

3.2 Model architectures

This section covers the different model architectures that we have implemented. These are five types of deep learning models: 1-dimensional CNN, 2-dimensional CNN, LSTM, regular self-attention and HAR self-attention. The final subsection will cover our active learning approach.

3.2.1 1D CNN

The first model implemented in this study is a 1-dimensional convolutional neural network. The architecture of the model is depicted in Figure 3.3 and consists of two convolutional layers with filter width equal to the number of columns in the windows of data (i.e., the number of features). This configuration allows the filter of the convolutional layer to "scan" the windows a few time steps at a time, thus convolving over the data in a sequential manner.
Batch normalization and dropout were added to stabilize the training process and prevent overfitting, respectively. The model was trained with the adaptive moment estimation (Adam) optimizer and with early stopping. The architecture shown in Figure 3.3 is the result of several experiments to find the best compromise between

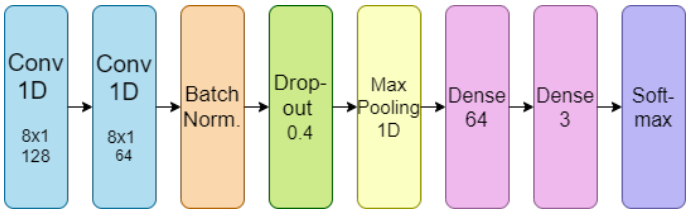


Figure 3.3: Architecture of the 1D CNN used in this study.

simplicity (and therefore, training speed) and performance: the model performed as well as more complex models, with faster training. The hyper-parameters of the 1D CNN have been tuned based on the model's performance on the validation set; the best configuration of hyper-parameters is reported in Table 4.12 of the Results chapter.

3.2.2 2D CNN

A time window can be converted into several recurrence plots. Each driving signal in the time window will be converted into a single recurrence plot according to Equation 2.27. Examples of this processing can be seen in Figure 3.6.

There are two approaches after the creation of recurrence plots. One of the approaches is to concatenate the recurrence plots into a single image. The other approach is to combine the recurrence plots into a *joint recurrence* plot using Equation 2.28. Both of these approaches have been adopted when training the 2D-CNNs on the test track data, whereas only the joint recurrence plot method has been adopted on the WICE data.

Tuneable hyper-parameters for the 2D-CNN are *filter size*, *amount of filters*, *amount of CNN layers*. Other parameters can also affect the results that are outside of model building. These parameters are related to generating the recurrence plots, e.g., the threshold in Equation 2.27. The best performing models are presented in the Results chapter with corresponding hyper-parameters.
The recurrence plots can thereafter be used as training data for the 2D-CNN.

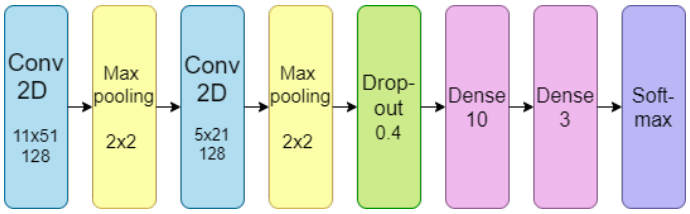


Figure 3.4: Architecture of the 2D CNN used in this study.

3.2.3 LSTM

Just like the 1D CNN, the architecture employed for the LSTM was the best compromise between simplicity and performance. The model layers are depicted in

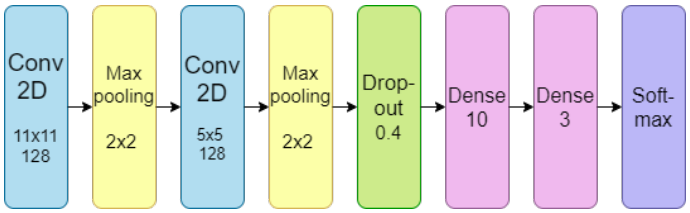
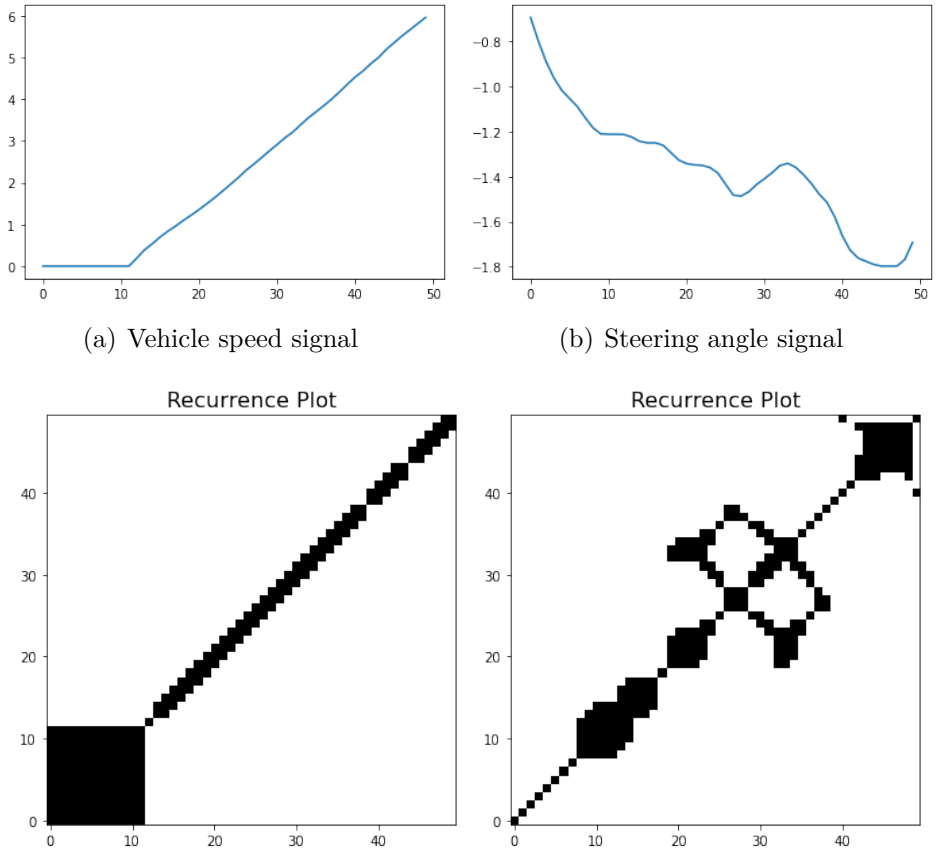


Figure 3.5: Architecture of the 2D CNN used in this study for joint recurrence.



(a) Vehicle speed signal

(b) Steering angle signal

(c) Recurrence plot of the signal in (a)

(d) Recurrence plot of the signal in (b)

Figure 3.6: Examples of processing driving signals into recurrence plots. Figure (a) and (b) shows examples of driving signals. Figure (c) and (d) show the corresponding recurrence plots computed from the each of the signals.

Figure 3.7: a one-directional LSTM layers is followed by a fully connected layer, followed by batch normalization and dropout to improve the training stability and prevent overfitting. The model was trained with the Adam optimizer and with early stopping. The hyper-parameters of the LSTM have been tuned based on the model's performance on the validation set; the best configuration of hyper-parameters is reported in Table 4.12 in the Results chapter.

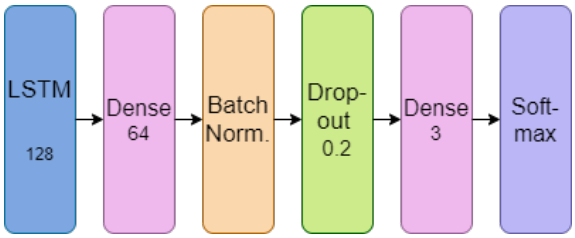


Figure 3.7: Architecture of the LSTM used in this study.

3.2.4 Self-attention networks

A network with multi-headed self-attention blocks (with positional encoding added to the inputs) was implemented as covered in subsection 2.1.4, together with a self-attention model as proposed by Mahmud et al. [39], from now on referred to as *HARSA* ("Human Activity Recognition self-attention"). The regular self-attention and HARSA models were trained and evaluated on the test track and WICE data, respectively, as each was found to perform slightly better than the other on the respective dataset.
The two types of self-attention models were trained on various hyper-parameter values. Hyper-parameters that are specific to self-attention are the number of heads, used in multi-head self-attention, and the number of self-attentive blocks (also known as the *depth* of a self-attention layer). Results of the best performing models with corresponding hyper-parameter values are shown in the Results chapter.

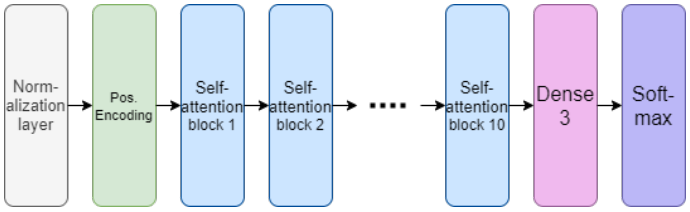


Figure 3.8: Architecture of the self-attention model used in this study.

3.2.5 Active learning

Three different active learning approaches have been explored: uncertainty sampling, query by committee, and gradual pseudo labeling (these are covered in section 2.3). A cumulative training approach has been employed to generate *learning curves*, which provide information about the performance of the active learner (see section 2.3 for details about learning curves). The cumulative training was implemented according to the procedure described by Bossér et al. [36]: first, the classifier was trained on the labeled set L; second, the unlabeled samples U were picked randomly or ranked according to the chosen informativeness measure, and batch $B \subseteq U$ of the n top samples was selected; finally, the classifier was re-trained on $L \cup B$ after parameter re-initialization [36]. This process was repeated until $|L \cup B|$ was sufficiently close to the size of the whole dataset.
All experiments were carried out on non-overlapping windows of 10 seconds on the

WICE data. The test set size was set to 20%; the test set was fixed throughout iterations of the same experiment. All experiments, except for uncertainty sampling on the HARSA model, were carried out 10 times with different random seeds. The experiments on the HARSA model were carried out with 3 random seeds.

3.2.5.1 Experiment I - uncertainty sampling

This experiment compared the performance of the models when trained with different informativeness measures of uncertainty sampling. We first trained the models on 10% of randomly picked data and then on 5% increments of samples selected through the chosen informativeness measure. We then assessed the test accuracy for 14 iterations, i.e., starting from 15% up to 80% of training data.

3.2.5.2 Experiment II - uncertainty sampling for time-series data

This experiment is based on the probability density function estimated for each of the five driving parameters described by Younes et al. [6] (see subsection 3.1.3). After computing the driving parameters for each window and determining their probability density function for each class from the test track data, we estimated the probability vectors of each window's driving parameter by class. We then computed the Jensen-Shannon distance between each pair of windows' probability vectors, and found 500 nearest neighbors and 200 reverse neighbors for each window. We incorporated this distance in both the uncertainty and the utility informativeness measure (to compute the similarity, in the latter case) proposed by Peng et al. [37], as described in section 2.4. We then trained the models on 5% increments of data, in the same way as with regular uncertainty sampling.

3.2.5.3 Experiment III - query by committee

For this experiment, we formed a committee of three members: the 1D CNN described in subsection 3.2.1, the LSTM described in subsection 3.2.3, and a CNN-LSTM whose architecture is depicted in Figure 3.9. Each committee member was trained on the labeled set and performed inference on the unlabeled set at each iteration. Both the vote entropy and the Kullback-Leibler divergence informativeness measures were tested, with 14 5% increments of training data.

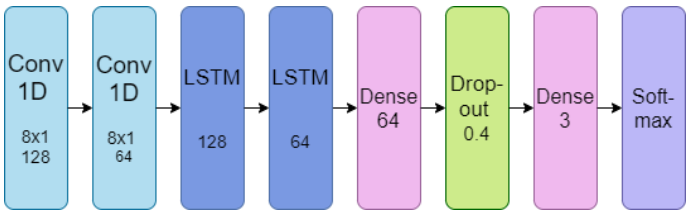


Figure 3.9: Architecture of the CNN-LSTM used as committee member.

3.2.5.4 Experiment IV - query by committee with active deep dropout

We implemented active deep dropout by training a *parent* model and then generating a committee of 5 members, where each member had the same architecture as the

parent but with a different dropout configuration. The dropout rate differed both between committee members and across iterations. The committee members performed inference on the unlabeled set through a single forward pass per iteration, and their predictions were used to estimate the vote entropy and the Kullback-Leibler divergence measures. The parent network was the re-trained on the dataset augmented with the batch of most informative samples. This procedure followed the optimization suggested by Gammelsæter [35], which avoids fully training all committee members at each iteration. The tested models were the 1D CNN, the LSTM, and the self-attention model, again with 5% increments of training data for 14 iterations.

3.2.5.5 Experiment V - gradual pseudo labeling

For this experiment, we first found the best confidence threshold ξ^* for each model by following the methodology proposed by Bossér et al. [36]: we first trained each model on 10% of randomly sampled data and then on further 10% of data queried randomly or through the margin informativeness measure.
Secondarily, we ranked the unlabeled samples according to the informativeness measure $I_i = -\max_y P_\theta(y|\boldsymbol{x_i})$ and we added all samples with $I_i < \xi$ to the confidence subset. Thirdly, if the size of the confidence subset was above threshold N, we generated pseudo labels for each sample as the class with the highest predicted probability, added the confidence subset to the labeled samples, and removed it from the unlabeled samples. Subsequently, we re-trained the model on the augmented labeled set, and repeated the process until the size of the confidence subset was equal to or smaller than N. Finally, we assessed the test accuracy and repeated the same process for 100 thresholds ξ between -1 and 0. We then fit a kNN regressor on the thresholds and corresponding test accuracies with $k = 10$; we picked the threshold yielding the highest test accuracy according to the regression fit as the best threshold ξ^*. We performed the experiment on the 1D CNN and the LSTM models and picked $N = 50$ for the former and $N = 20$ for the latter.
After finding ξ^*, we generated a learning curve by iteratively selecting and generating pseudo labels for the confidence subset of the samples with $I_i < \xi^*$; the confidence subset size was limited to 5% of the whole dataset per iteration as in the previous experiments.

4
Results

This chapter reports the performance of the models in the classification task (section 4.1), as well as the result of applying the active learning methods (section 4.2).

4.1 Classification

This section reports the models' performance on the test track and WICE data, across several experiments.

4.1.1 Test track

Table 4.1 and Table 4.2 report the results gained from data gathered on the test track. More specifically, these tables show the performance amongst the models and their performance on each of the driver behavior classes. Table 4.3 shows the overall performance of the models using metrics such as accuracy and macro-averaged precision, recall, F_1 score, and AUC. Table 4.4 shows the parameter size of each model.

Insight from these results will be covered in the later section 5.1.3.1.

	LSTM			**1D CNN**		
Class	**Precision**	**Recall**	F_1 **score**	**Precision**	**Recall**	F_1 **score**
Aggressive	0.983	0.933	0.957	0.987	0.895	0.943
Normal	0.861	0.538	0.665	0.892	0.413	0.565
Cautious	0.673	0.935	0.777	0.611	0.952	0.745

Table 4.1: Classification results for the LSTM and 1D CNN models.

	Self-attention			**2D CNN**		
Class	**Precision**	**Recall**	F_1 **score**	**Precision**	**Recall**	F_1 **score**
Aggressive	0.981	0.941	0.963	0.382	0.484	0.423
Normal	0.451	0.152	0.227	0.502	0.275	0.355
Cautious	0.612	0.902	0.733	0.463	0.723	0.575

Table 4.2: Classification results for the self-attention and the 2D CNN models.

Model	Accuracy	Macro-avg Precision	Macro-avg Recall	AUC
LSTM	**0.825**	0.847	0.803	**0.952**
1D CNN	0.783	0.835	0.753	0.916
Self-attention	0.733	0.685	0.664	0.884
2D CNN	0.445	0.454	0.496	0.697

Table 4.3: Results for all of the models on the test-track data.

Model	Total parameters	Non-trainable parameters
LSTM	77 251	128
1D CNN	65 155	128
Self-attention	3 993	0
2D CNN	117 303	0

Table 4.4: Number of parameters per model.

4.1.2 WICE

Table 4.5 reports the average evaluation metrics of the four implemented models on 10-second windows extracted from WICE. As mentioned in subsection 3.1.4, the stability in performance of the models has been assessed through stratified 5-fold validation. For this dataset, the accuracy of a majority-class classifier is $\approx 36.5\%$.

Model	Accuracy	Accuracy σ	Weighted-avg AUC
LSTM	0.846	0.026	0.951
1D CNN	**0.881**	0.03	**0.972**
HARSA	0.694	0.025	0.822
2D CNN	0.602	0.083	0.774

Model	Weighted-avg Precision	Weighted-avg Recall	Weighted-avg F_1 score
LSTM	0.847	0.846	0.844
1D CNN	0.882	0.88	0.88
HARSA	0.734	0.732	0.729
2D CNN	0.609	0.602	0.597

Table 4.5: Models' performance on the WICE data, with 10-second non-overlapping windows.

All models report a relatively low standard deviation (σ) of the accuracy, which indicates good robustness across different training-test splits. They also report similar precision and recall.

Table 4.6 reports the confusion matrix of an LSTM run that yielded an average accuracy. Perhaps unsurprisingly, most of the classification error occurred for the normal class, accounting for more than 88% of misclassifications when considering both false negatives and false positives. Table 4.7, the confusion matrix for a 1D

CNN run with average accuracy, reports similar findings, with 84% of misclassifications involving the normal class. This was expected, as the mean of the driving parameters proposed by Younes et al. for the normal class lies between the other two classes' parameter distribution mean.

Actual \ Predicted	Cautious	Normal	Aggressive
Cautious	229	17	3
Normal	30	188	31
Aggressive	8	9	167

Table 4.6: Confusion matrix for LSTM.

Table 4.8 reports weighted-averaged precision, recall, and F_1 score per class for the LSTM and 1D CNN models on the 10-second windows dataset. The LSTM shows a considerably higher precision than recall for the normal class, and the opposite situation for the cautious class. This likely means that the classifier had a slight tendency to mislabel normal samples as cautious, even though the available results are not enough to determine if this difference is statistically significant.

Actual \ Predicted	Cautious	Normal	Aggressive
Cautious	237	8	4
Normal	16	201	32
Aggressive	8	8	168

Table 4.7: Confusion matrix for the 1D CNN.

	LSTM			1D CNN		
Class	**Precision**	**Recall**	F_1 **score**	**Precision**	**Recall**	F_1 **score**
Aggressive	0.839	0.846	0.841	0.848	0.89	0.868
Normal	0.856	0.763	0.806	0.889	0.828	0.856
Cautious	0.845	0.928	0.884	0.901	0.927	0.913

Table 4.8: Precision, recall and F_1 score by class for the LSTM and 1D CNN models.

Table 4.9 reports the performance metrics for the three models on 5-second, non-overlapping windows. As for the 10-second windows, 5-fold cross validation was applied to verify the robustness of the models. For this dataset, the accuracy of a majority-class classifier is $\approx 34.9\%$.

Model	Accuracy	Accuracy σ	Weighted-avg AUC
LSTM	0.863	0.012	0.962
1D CNN	**0.889**	0.015	**0.976**
HARSA	0.652	0.04	0.777
2D CNN	0.543	0.045	0.721
Model	**Weighted-avg Precision**	**Weighted-avg Recall**	**Weighted-avg F_1 score**
LSTM	0.864	0.863	0.862
1D CNN	0.889	0.889	0.888
HARSA	0.774	0.772	0.766
2D CNN	0.545	0.543	0.541

Table 4.9: Models' performance on the WICE data, with 5-second non-overlapping windows.

Table 4.10 reports the models' performance on the WICE data, with 5-second windows with 50% overlapping. We evaluated the models on a hold-out test set, instead of performing cross-validation, to prevent data leakage across sets. We tested the models on 5 random seeds. The accuracy of a majority-class classifier is $\approx 35.9\%$ in this case.

Model	Accuracy	Accuracy σ	Weighted-avg AUC
LSTM	0.832	0.009	0.947
1D CNN	**0.863**	0.006	**0.964**
HARSA	0.745	0.012	0.887
2D CNN	0.46	0.004	0.653
Model	**Weighted-avg Precision**	**Weighted-avg Recall**	**Weighted-avg F_1 score**
LSTM	0.833	0.832	0.831
1D CNN	0.865	0.863	0.863
HARSA	0.797	0.795	0.793
2D CNN	0.47	0.46	0.457

Table 4.10: Models' performance on the WICE data, with 5-second overlapping windows.

The three experiments consistently show the 1D CNN as the best performing model for all reported metrics, and 2D CNN as the worst performing model. They also do not show a stark difference between precision and recall, meaning that the models present a generally good balance between the number of samples classified as positive for each class and the correctness of this classification. The robustness of the models, quantified through the standard deviation of the accuracy, generally increases across the three experiments, with the highest standard deviation found with 10-second windows and the lowest found with overlapping 5-second windows. The fact that 10-second windows yield more variable results is consistent with the lower annotation granularity of larger windows: the larger a window is, in fact, the higher the chance that it presents different driving styles. Table 4.11 compares the number of trainable and non-trainable parameters across the models: unsurprisingly,

performance correlates with number of parameters (with the exception of the 2D CNN networks - we see a Pearson correlation of 0.931 for the test accuracy of the LSTM, 1D CNN, and HARSA model in the experiment with overlapping windows).

Model	Total parameters	Non-trainable parameters
LSTM	78 787	128
1D CNN 5-second windows	176 771	128
1D CNN 10-second windows	279 171	128
HARSA model	11 131	0
2D CNN 5-second windows	35 603	0
2D CNN 10-second windows	131 803	0

Table 4.11: Number of parameters per model.

Table 4.12 and Table 4.13 list the hyper-parameters tested on the LSTM and the 1D CNN and on the the HARSA model and the 2D CNN, respectively. The values included in the final version of the models are shown in bold. The hyper-parameters have been tuned based on performance on the validation set of the 10-second windows WICE dataset.

4.2 Active learning

Figure 4.1 compares the learning curves produced by several uncertainty sampling techniques and random sampling on the LSTM model; Figure 4.2 shows the same comparison for the 1D CNN model, and Figure 4.3 shows the comparison without ACTS for the HARSA model.

We can notice a considerable performance variability across uncertainty sampling methods and models. Margin, entropy, and least confidence perform quite reliably better than random sampling for the LSTM and the CNN, although random sampling yields a higher test accuracy in the early iterations for the LSTM than margin and entropy. ACTS appears to have improved the training process only with the CNN, both with 10 and 30 neighbors, and not with the LSTM. The HAR self-attention model does not seem to have benefited from active learning.

LSTM		1D CNN	
Hyper-parameter	**Tested values**	**Hyper-parameter**	**Tested values**
Nr. of neurons in hidden LSTM layer	32, 64 **128**, 256	Nr. of filters in 1st CNN layer	64, **128**
Dropout rate	None, **0.2**, 0.3, 0.5	Nr. of filters in 2nd CNN layer	**64**, 128
LSTM dropout rate	None, **0.2**, 0.3	First dropout rate	None, 0.2, 0.3, **0.4**
Number of LSTM layers	**1**, 2, 3	Second dropout rate	None, 0.2, 0.3, **0.4**
Activation function in hidden dense layer	relu, **tanh**	Nr. of neurons in hidden dense layer	32, **64**
Batch size	64, **128**	Batch size	64, **128**
Learning rate	**0.001**, 0.005, 0.01	Learning rate	**0.001**, 0.005, 0.01
Early-stopping patience	10, 20, **30**	Early-stopping patience	10, 20, **30**

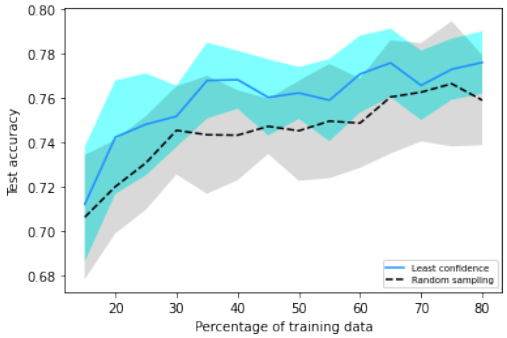
Table 4.12: Hyper-parameters tested on the LSTM and the 1D CNN. The values in the final models are in bold.

Figure 4.4 and 4.5 compare the performance of the LSTM and 1D CNN models, when trained with random sampling and with query by committee techniques. The two classifiers present diverging results: regular query by committee methods were successful for the CNN but not for the LSTM, while active deep dropout methods worked successfully on the LSTM but not on the CNN, except for the earliest iterations.

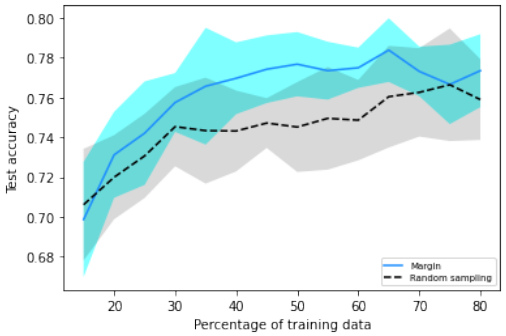
2D CNN		HARSA	
Hyper-parameter	**Tested values**	**Hyper-parameter**	**Tested values**
Nr. of filters in CNN layers	**20**, 50	Nr. of heads in multi-head attention	4, **8**
% of points < threshold for min. distance in joint recurrence plots	25, **50**	Depth	10, **15**, 30
Dropout rate	0.3, **0.5**	Nr. of filters in 2D CNN layers	10, 50, **100**
Batch size	64, **128**	Batch size	64, **128**
Learning rate	**0.001**, 0.01	Learning rate	**0.001**, 0.005, 0.01
Early-stopping patience	**5**, 10, 30	Early-stopping patience	20, **30**

Table 4.13: Hyper-parameters tested on the 2D CNN and HAR self-attention models. The values in the final models are in bold.

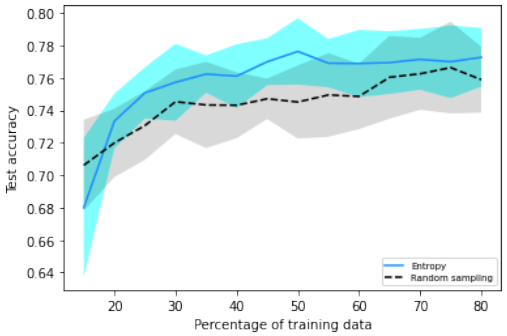
Figure 4.6 shows the kNN regression fit with $k = 10$ for 100 thresholds $\xi \in [-1, 0]$ for the LSTM and 1D CNN models. Similarly to the findings of Bossér et al. [36], we can observe an almost negative correlation between threshold and test accuracy for the LSTM, and a negative correlation when $\xi \geq -0.52$ for the CNN. As shown in Figure 4.7, the GPLA random and margin techniques have not been able to perform better than random sampling for the same percentage of training data, except for the early iterations on the CNN. The test accuracy overall remains quite stable or even slightly decreases as the training data increases, suggesting that the samples labeled through semi-supervised learning do not carry relevant information from which the classifiers can learn.



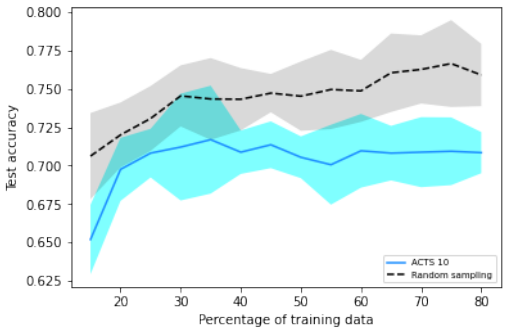
(a) Least confidence



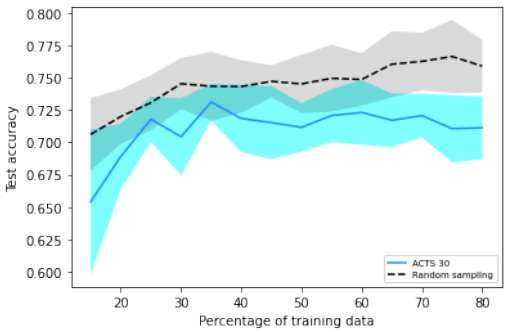
(b) Margin



(c) Entropy

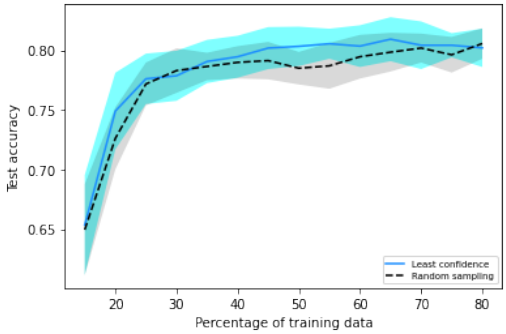


(d) ACTS 10 neighbors

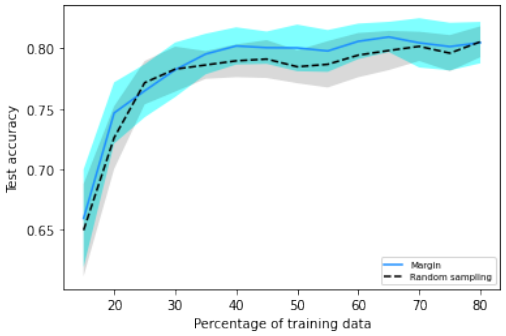


(e) ACTS 30 neighbors

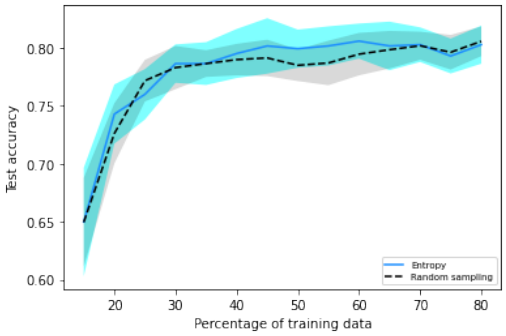
Figure 4.1: Comparison of uncertainty sampling methods with random sampling on the LSTM model. The solid line indicates the mean test accuracy for each iteration across 10 experiments with different random seeds, while the transparent area around the line represents the margin covered by the mean ± the standard deviation.



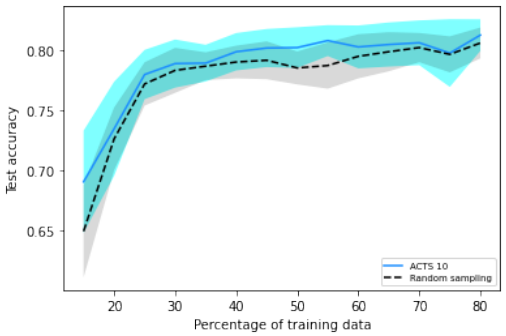
(a) Least confidence



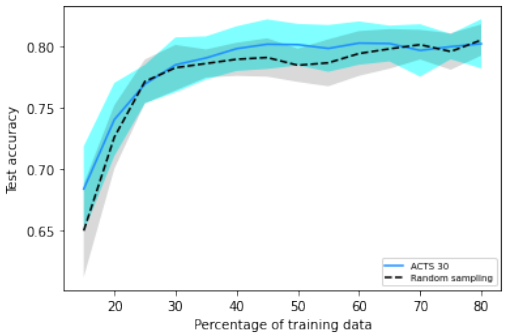
(b) Margin



(c) Entropy

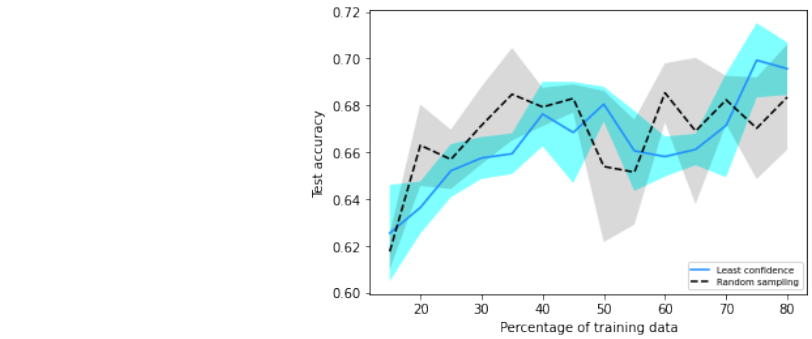


(d) ACTS 10 neighbors

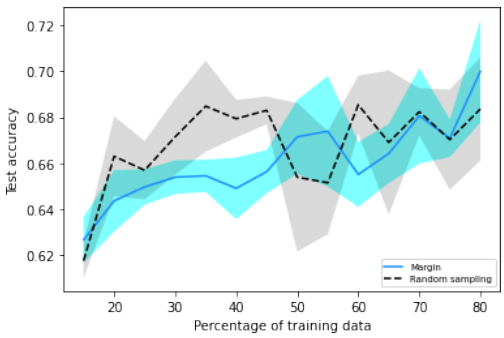


(e) ACTS 30 neighbors

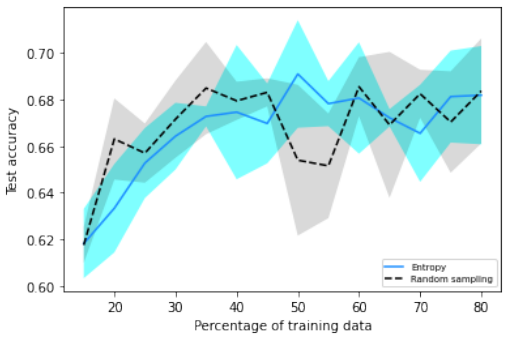
Figure 4.2: Comparison of uncertainty sampling methods with random sampling on the 1D CNN model. The solid line indicates the mean test accuracy for each iteration across 10 experiments with different random seeds, while the transparent area around the line represents the margin covered by the mean ± the standard deviation.



(a) Least confidence

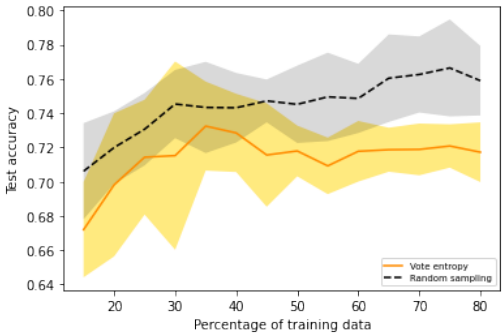


(b) Margin

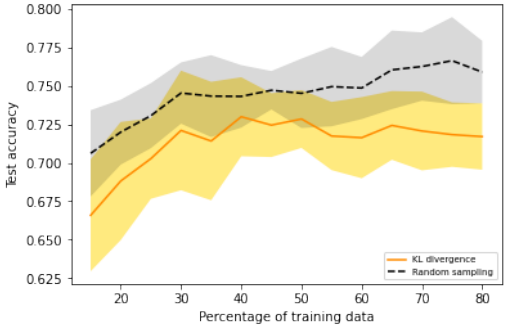


(c) Entropy

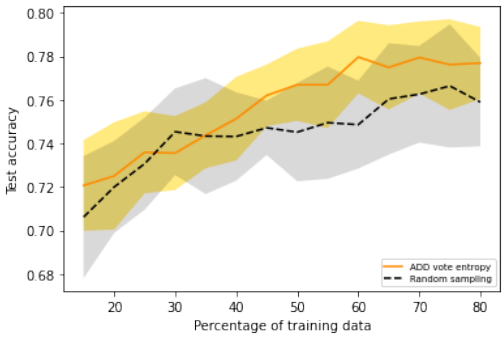
Figure 4.3: Comparison of uncertainty sampling methods with random sampling on the HAR self-attention model. The solid line indicates the mean test accuracy for each iteration across 3 experiments with different random seeds, while the transparent area around the line represents the margin covered by the mean ± the standard deviation.



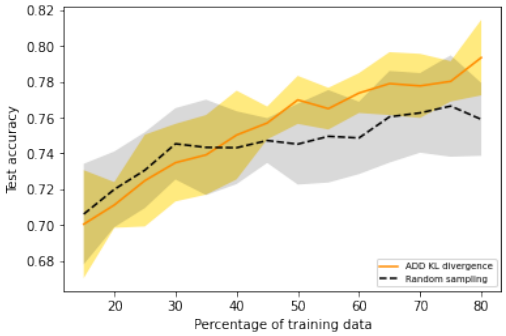
(a) Vote entropy



(b) KL divergence

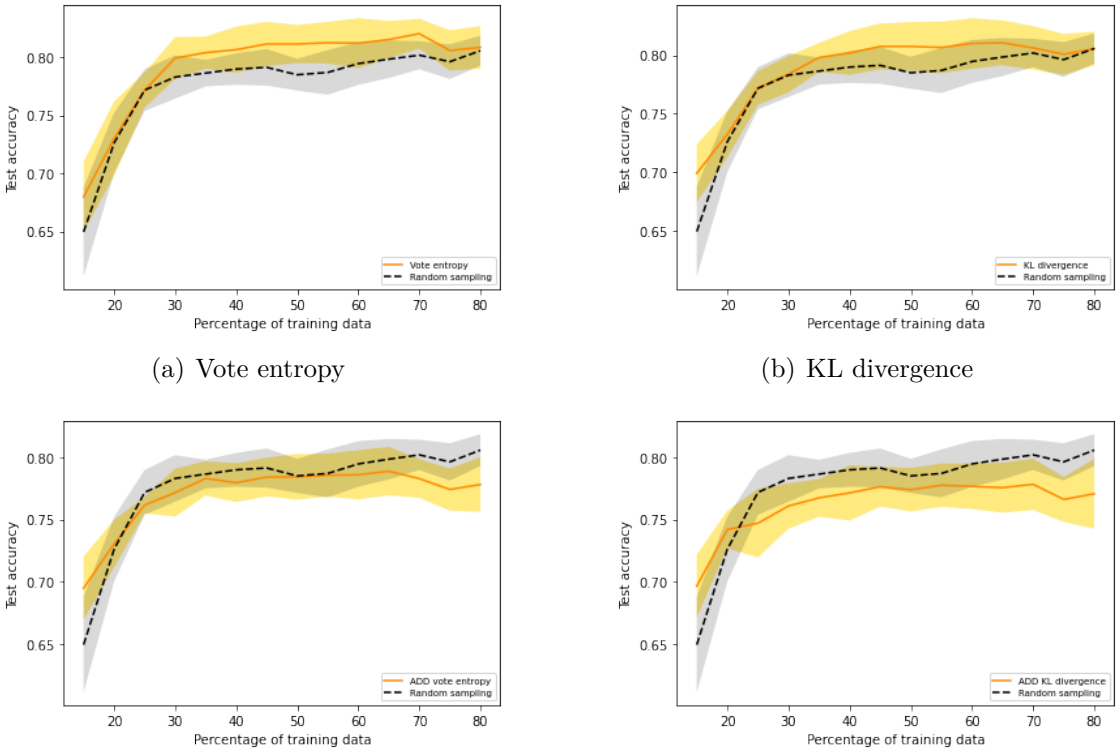


(c) ADD vote entropy



(d) ADD KL divergence

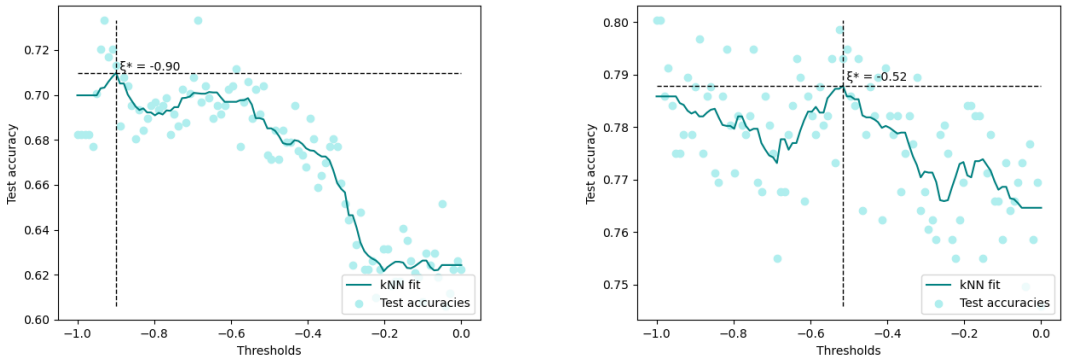
Figure 4.4: Comparison of query by committee methods with random sampling on the LSTM model. The solid line indicates the mean test accuracy for each iteration across 10 experiments with different random seeds, while the transparent area around the line represents the margin covered by the mean ± the standard deviation.



(a) Vote entropy (b) KL divergence

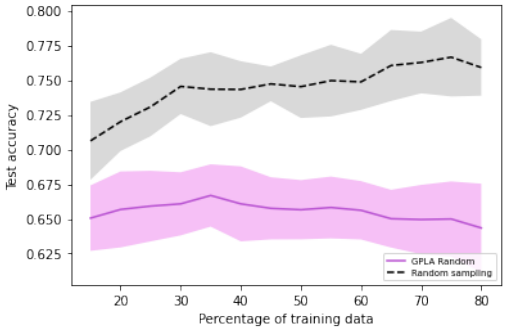
(c) ADD vote entropy (d) ADD KL divergence

Figure 4.5: Comparison of query by committee methods with random sampling on the 1D CNN model. The solid line indicates the mean test accuracy for each iteration across 10 experiments with different random seeds, while the transparent area around the line represents the margin covered by the mean ± the standard deviation.

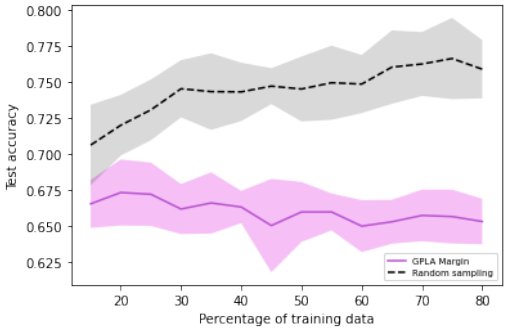


(a) LSTM best threshold (b) CNN best threshold

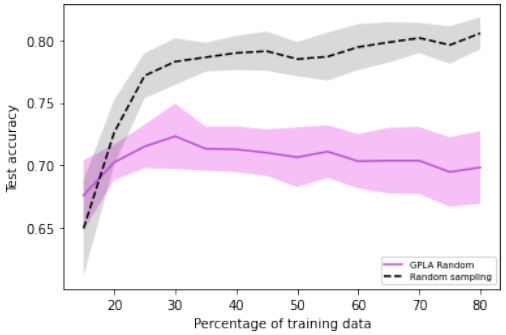
Figure 4.6: Best confidence thresholds for (a) the LSTM and (b) the 1D CNN.



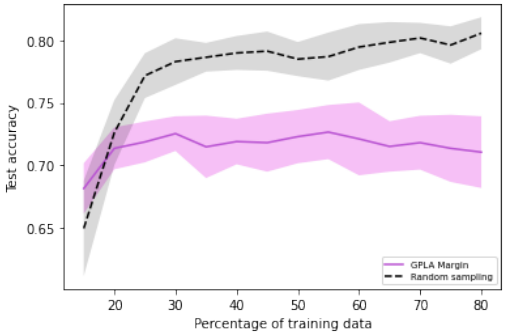
(a) LSTM GPLA random



(b) LSTM GPLA margin



(c) CNN GPLA random



(d) CNN GPLA margin

Figure 4.7: Comparison of GPLA methods with random sampling on (a) and (b) the LSTM model and (c) and (d) the 1D CNN model. The solid line indicates the mean test accuracy for each iteration across 10 experiments with different random seeds, while the transparent area around the line represents the margin covered by the mean ± the standard deviation.

5

Discussion and Conclusion

This final section of the thesis is broken down into three parts: discussion (section 5.1), conclusion (section 5.2), and future work (section 5.3). These sections will cover the quality of our research, conclusions from the results, and suggestions for future work.

5.1 Discussion

Our research objectives were stated in section 1.3. The quality of our research can be analyzed in terms of *validity* and *reliability*, which will be discussed in the following two subsections. The last subsection covers the major findings from our results.

5.1.1 Validity

Our overall research objective is to model driving behavior. One validity issue is the ambiguity of this term. This ambiguity specifically affects our recreation of driving styles on the test track (as described in subsection 3.1.1). Previous research presents informal descriptions of driving styles, such as "*abnormal and immediate changes in vehicle speed, inappropriate keeping of vehicle lateral position, hazardous lane change, and fast acceleration and deceleration takeoff and braking*" [10]. However, these descriptions are non-numerical and are hard to re-create without relying on intuition. This creates a validity issue for our generated data on the test track. We have tried to mitigate this issue by re-creating the driving styles according to previous work [6, 10, 26] with several driving trials before gathering data. Overall, however, the definition of aggressive driving is quite subjective and dependent on culture. This ambiguity and subjectivity makes it hard to establish a universally valid ground truth with which the obtained results can be compared.

5.1.2 Reliability

We will cover two reliability issues: external factors affecting driving behavior, and stochastic elements in model building.

Our first reliability issue is trying to control external factors that can affect driver behavior. These factors are framed in the conceptual framework (see section 1.1).

There are three main modules affecting driver behavior: driver, vehicle, and environment. The following list breaks down these modules in more detail:

- **Driver module:** includes driver profile such as driving experience, age, etc.
- **Vehicle module:** includes gadgets and applications in/outside the vehicle such as parking monitors, cruise control, etc. The vehicle model itself can affect the driving experience of each driver
- **Environment:** these include road condition, weather condition, and traffic condition.

Most of these factors were easily controlled when generating data on the test track: we generated our data on the same road (the test track shown in Figure 3.1) with similar weather and traffic conditions. However, it can be argued that the test track data does not capture driving styles from a wide variety of drivers. This is mainly because we have only collaborated with two drivers who simulated different driving styles. Due to this reason, we have also experimented our analysis on the WICE data that contains several drivers. We have also controlled for factors as road condition and vehicle model when analyzing the WICE dataset. However, we could not fully control for weather condition.

Our second reliability issue comes from the stochastic elements when building the models. There are two stochastic elements to consider: training and test split and the initialization of the models. The first issue affects the performance of the model based on how similar the test set is to the training set; it can be addressed by using k-fold cross validation. The second issue affects the training progress as different initialization values find different local minima of the loss function. This can easily be bypassed by re-training the models with different random seeds. This stochasticity is relevant for our project because we have employed relatively small datasets. Larger datasets, however, would allow a more stable performance regardless of initialization and dataset split.

5.1.3 Discussion of findings

This section will cover major findings found from the result section (chapter 4) and try to put them into previous research context. It is divided into two subsections: results of the driver behavior classification and results of the active learning experiments.

5.1.3.1 Driver behavior classification results

Table 4.1 and Table 4.2 show that the majority of the models manage to recognize the aggressive class on the test track data. The 2D CNN did not perform nearly as well, and is only slightly better than a dummy classifier (i.e., majority class or random classifier). There are several reasons that can be speculated as to why this is the case.

The 2D CNN pre-processes the driving signals into recurrence plots according to Equation 2.27. Given one signal of n values, the corresponding recurrence plots would have size $n \times n$. This processing of the driving signals creates larger input

space to be modeled requiring more data. We speculate that the 2D CNN requires more data to increase its performance. The benefits of using spatial information in the recurrence plots is its ability to bypass sequential training [10].

Some final remarks from subsection 4.1.1 and subsection 4.1.2 can be noted about the model complexity shown in Table 4.4 and Table 4.11. The self-attention model did perform slightly worse than the 1D CNN and LSTM as seen in Table 4.2. However, it can also be noticed from Table 4.4 that the self-attention model has a lower model complexity than the rest. Both LSTM and 1D CNN have more than 16 times the amount of parameters of the self-attention model. This is beneficial in terms of memory and computational efficiency when using machine learning models on smaller computer architectures, such as micro-controllers or micro-computers. Self-attention has also the benefit of training on windows with parallel computation.

We also believe that the geometry of the test track shown in Figure 3.1 forces some restrictions on the driving performance. The smooth corner seen on the upper side of the figure, which is a considerable portion of the test track, prevents the drivers from performing abrupt manoeuvres, thus forcing them to adopt a cautious or normal driving style. We speculate that this limits the maximum accuracy achievable for classification on the test track data.

The results of the models on the WICE data show that all models have learned to classify driver behavior, albeit with considerable difference across models and across window size. The 1D CNN model consistently yielded the highest accuracy and AUC, while also presenting the highest number of parameters. The 1D CNN and the LSTM showed a higher accuracy on non-overlapping windows (of either 10 or 5 seconds), while the HARSA model performed better on overlapping windows. The difference in performance for HARSA might suggest a greater sensitivity of the model to the size of the training data: the 5-second, non-overlapping windows dataset (on which HARSA performs the worst) includes in fact 254 500 time steps in total, while the 10-second windows dataset includes 341 200, and the 5-second overlapping windows dataset includes 496 750. Furthermore, the lower number of parameters of the HARSA model might be the reason of its lower accuracy throughout experiments: HARSA yields the highest accuracy per number of parameters. The 2D CNN overall failed to classify with appreciable results, yielding an accuracy slightly higher than that of the majority-class classification for 5-second overlapping windows. One reason for this might be that the generated joint recurrence plots did not include enough meaningful information for the classifier to distinguish correctly between the three classes. Further experimentation with joint recurrence plots hyper-parameters might have generated more meaningful plots. Another reason might be the loss of information due to the Hadamard product performed by joint recurrence: stacking separate plots for each signal, instead of joining them through the Hadamard product, might have produced better results.

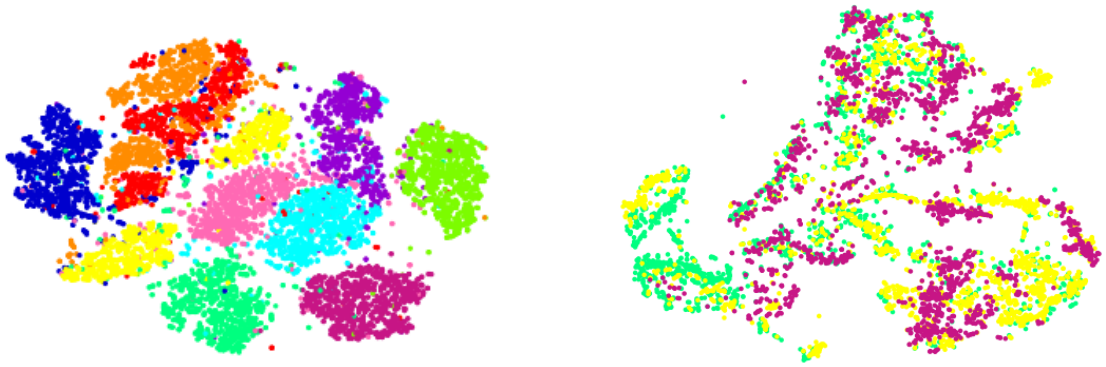
When compared to previous work, even the best performing models (i.e., the 1D CNN and the LSTM) fail to fall above the lower quartile for accuracy or the median

for recall when compared to the 47 studies reviewed by Elamrani Abou Elassad et al. [7]. However, this comparison may need to be taken with reservation: many of those studies, in fact, perform tasks on data presenting a higher granularity than time windows. One such example is the detection of abnormal driving events, e.g. swerving and fast U-turn manoeuvres [58]. Also, driving simulators (as shown in [59] and [60]) may allow for riskier behavior than naturalistic driving, thus producing starker differences in the driving feature distribution among classes.

5.1.3.2 Active learning

Our active learning simulations show wide variety of results. LSTM and CNN performed best with more conventional active learning methods, such as uncertainty sampling and query by committee (seen in Figure 4.1 and Figure 4.2). Out of our novel active learning approaches, only ADD with LSTM and ACTS with CNN outperformed random sampling. Unfortunately, GPLA did not show an improvement (seen in Figure 4.7) and neither did the active learning approaches with the self-attention architecture (seen in Figure 4.3).

It can be speculated that the GPLA did not perform as well as other methods because the generated pseudo labels did not provide the model with enough helpful information to learn effectively. The difference with the results reported by Bossér et al. might be due to the use of different datasets for the experiments and, consequently, the effect of the size of the initial training data on the capability of the models to perform accurate predictions. The authors performed the experiments on the MNIST handwritten digits dataset, which shows a considerably higher inter-class separability than the WICE data, as seen after dimensionality reduction with 2-component tSNE in Figure 5.1. The higher inter-class separability of the MNIST



(a) (b)

Figure 5.1: Results of 2-component tSNE on (a) 10 000 samples from MNIST and (b) 5-second, non-overlapping windows from WICE.

dataset, together with its considerably larger size (60 000 samples), likely meant that the 1.6% of the samples employed as initial training set by Bossér et al. provided more useful information to the classifiers than our initial 20% training data. Furthermore, unlike MNIST, the WICE dataset does not appear to support the

semi-supervised smoothness assumption and the *cluster assumption* that Chapelle, Schölkopf and Zien [61] report as necessary for semi-supervised learning to produce more accurate predictions than a classifier trained exclusively on labeled data with supervised learning. The authors define the smoothness assumption as:

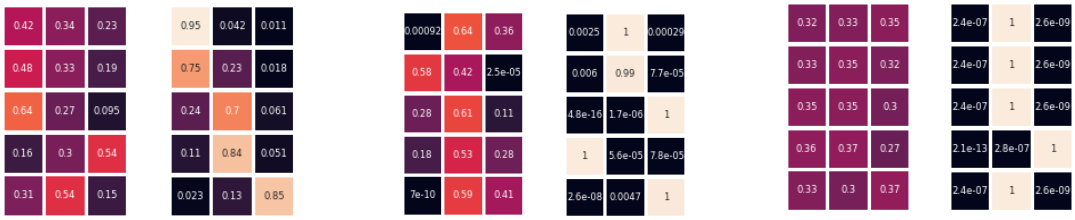
> *If two points x_1, x_2 in a high-density region are close, then so should be the corresponding outputs y_1, y_2.*

We can see from Figure 5.1b that this not often the case for the WICE data, as high-density regions of the bidimensional space often include points from two or more classes. The authors define the cluster assumption as:

> *If points are in the same cluster, they are likely to be of the same class.*

Again, we can observe that this is not often true for the clusters seen in Figure 5.1b: the samples do not appear to present a well-defined cluster structure which might justify semi-supervised learning. Overall, the WICE dataset does not appear to lend itself well to this kind of technique. However, the GPLA strategies may have shown an appreciable level of test accuracy (over 65% and 70% for the LSTM and CNN, respectively) when considering that they required only 10% of training data, i.e., 2740 seconds of driving data.

ACTS has yielded successful results on the 1D CNN, and unsuccessful results on the LSTM. In the former, it outperforms uncertainty sampling with entropy, which it is based upon, for approximately 78% of the iterations with 10 neighbors and 57% with 30 neighbors. Figure 5.2a and Figure 5.2b show the heatmaps of the 5 top (left) and 5 bottom (right) predictions of the LSTM and 1D CNN models, according to the uncertainty and utility informativeness measure. Figure 5.2c shows a heatmap of the predictions when using the entropy informativeness measure on the 1D CNN. Each row represents one sample and each column represents a label. A uniform color across the labels suggests that model has output similar predictions



(a) (b) (c)

Figure 5.2: Heatmaps of the 5 top and 5 bottom samples according to uncertainty and utility, for (a) the LSTM and (b) the 1D CNN, and according to (c) entropy on the 1D CNN.

for each class, and therefore it is uncertain on the true label of each sample. This is particularly visible for the top samples with the entropy informativeness measure, in Figure 5.2c (left). A less uniform color suggests, instead, that the model is fairly

confident in its predictions. This is particularly visible for the bottom samples in Figure 5.2b and Figure 5.2c. Overall, we have observed that the top samples ranked by the uncertainty and utility informativeness measure do not always coincide with the samples about which the model is most uncertain. We can speculate that this has a beneficial effect on the 1D CNN but not on the LSTM, which has seen a considerable improvement in performance through the regular uncertainty sampling techniques. We can also observe from Figure 4.1d and Figure 4.1e that the accuracy of active learning with ACTS on the LSTM increases as the number of neighbors increases (this can be compared to the performance of ACTS with 5 neighbors shown in Figure 5.3). Further increasing the number of neighbors might make the ACTS queries more informative for the LSTM.

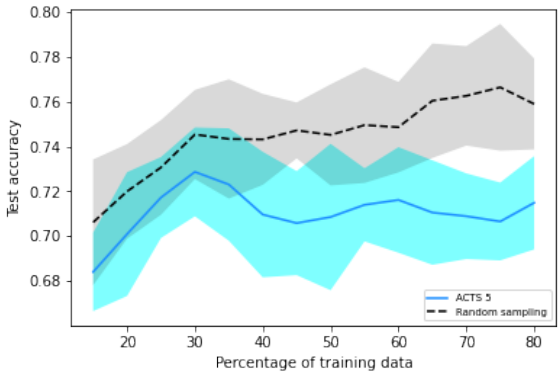


Figure 5.3: Results of ACTS with 5 neighbors on the LSTM model.

Active deep dropout showed very promising results for the LSTM (see Figure 4.4c and Figure 4.4d), especially when compared to regular query by committee (see Figure 4.4a and Figure 4.4b). The opposite situation can be encountered for the 1D CNN model. We speculate that the predictions of the LSTM might have been meaningful for the CNN (and therefore, active learning on the CNN yielded better results when the LSTM was included as a committee member), whereas the predictions of the CNN might have been unhelpful for the LSTM. This is consistent with the findings of Lowell et al. (2019) [62], who found a strong coupling between acquired training sets and the model with which they were acquired.

The variability of the results is also consistent with the results reported by Lowell et al.: they state that the performance of active learning strategies seems to be quite unstable and to vary strongly depending on the dataset and design choices. They also mention that experimentation might be necessary to find a query strategy that will perform considerably better than random sampling.

5.1.4 Correlation with energy consumption

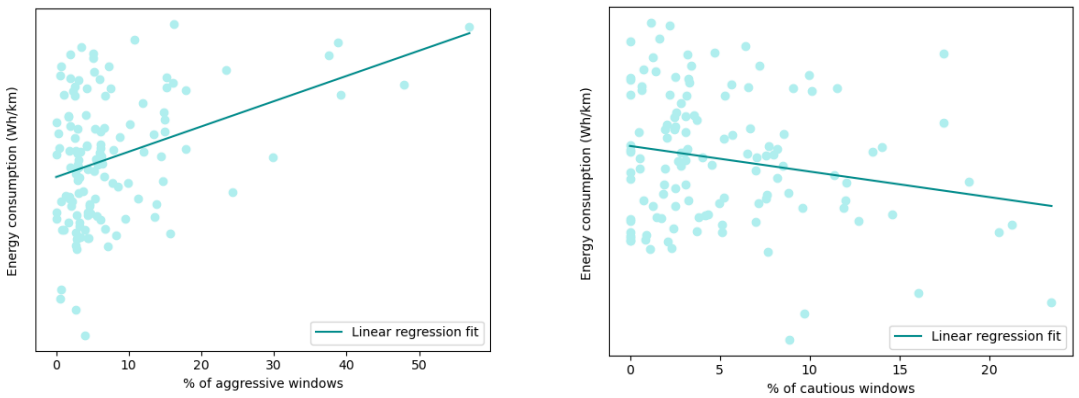
We investigated the correlation between the annotation resulting from the methods discussed in subsection 3.1.3 and the energy consumption of the trips. Since instantaneous energy consumption can vary greatly depending on traffic conditions, and the regenerative braking system in BEVs has a recharging effect on the battery, we

opted to compute the energy consumption for whole trips on Hisingsleden, instead of single windows. We calculated the average energy consumption for a trip as a function of the instantaneous battery voltage, the instantaneous battery current, and the distance covered during the trip. The consumption was measured in Wh/km. We then computed the number of aggressive, normal, and cautious windows and obtained the corresponding class percentage for each trip. Finally, we computed the Pearson correlation between the percentage of aggressive and cautious windows per trip and the trip total energy consumption: the resulting Pearson coefficients and two-tailed p-values for testing non-correlation are listed in Table 5.1.

Class	Pearson coefficient	p-value
Aggressive	0.368	0.00002
Cautious	-0.197	0.026

Table 5.1: Pearson coefficients and p-values for the correlation between annotation and energy consumption.

We can see that the percentage of cautious windows correlates negatively with energy consumption, while the percentage of aggressive windows correlates positively. This is not surprising and it is in the line with the findings of Bingham et al. [5] and Xing et al. [9]. The proposed annotation method can therefore be meaningful in the estimation of energy consumption in BEVs.
Figure 5.4 shows the scatter plot of the energy consumption by percentage of aggressive and cautious windows, as well as the linear regression fit for the data. A clear positive and negative trend can be observed depending on the prevalence of aggressive and cautious windows, respectively.



(a) (b)

Figure 5.4: Scatter plots and regression fit for energy consumption by percentage of (a) aggressive and (b) cautious windows.

From the scatter plots, however, we can notice a large number of trips with a low percentage of aggressive or cautious windows and also a large variance in energy consumption. There is in fact a considerable overlap between the three classes: if

we select the 50 trips with the highest percentage of aggressive, normal, and cautious windows, and plot a histogram of the energy consumption for the three groups, we can notice just that (see Figure 5.5). The distribution of the energy consumption by class can be compared to that of the RMSPF and jerk mean, shown in Figure 5.6.

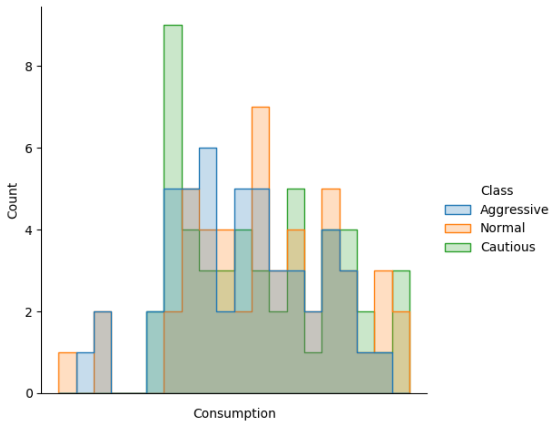
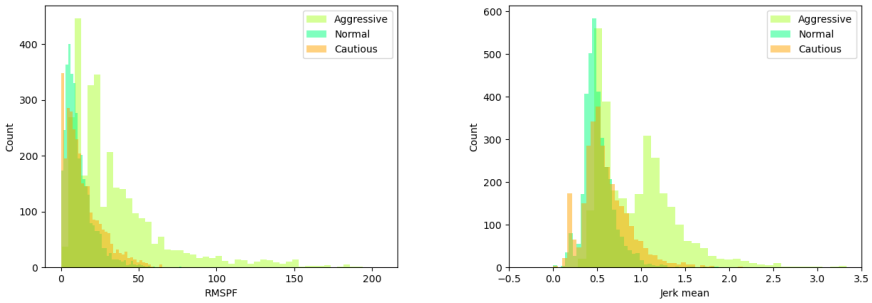


Figure 5.5: Histogram of the energy consumption by class.



(a) RMSPF (b) Jerk mean

Figure 5.6: Histograms of (a) RMSPF and (b) jerk mean by class.

The overlap observed for energy consumption might be partly due to the short length of the trips considered, as most of the trips collected are below 10 km. A larger inter-class separability in energy consumption might be expected for longer trips. Also, regenerative braking might have a smoothing effect on the energy consumption of aggressive drivers, since, according to our annotation method and findings in previous work [26, 29], they tend to brake more abruptly. Finally, the overlapping between classes might be a consequence of defining a frequent low gap time with the vehicle in the front as aggressive behavior: driving close behind a large vehicle for prolonged time periods, in fact, might lower the air drag and therefore yield a lower energy consumption for the tailgating vehicle.

5.2 Conclusion

Our study has employed various deep learning architectures to model aggressive driver behavior using passive and active learning. We relate our results to the first research question:

> *Can deep learning architectures achieve good accuracy on the driver behavior classification problem?*

We have determined that some of the proposed deep learning architectures have successfully learned to detect three different driving styles, both on data generated on a test track and on real-life data. Throughout experiments, the best performing models were the LSTM and 1D CNN. Our self-attention based models (Figure 3.8 and Figure 2.4) only performed slightly worse than LSTM and 1D CNN on test track data, and significantly worse on a few experiments on the WICE data. However, the parameter size of the self-attention model is considerably smaller than that of the LSTM and 1D CNN. The worst performing model is the 2D CNN. Shahverdy et al. [10] implemented similar 2D CNN models using recurrence plots with more promising results [10]. We speculate that our results do not match due to a difference in data processing.

The results of our experiments with active learning techniques can be used to answer our second research question:

> *Can the investigated active learning approaches achieve good accuracy with less labeled data?*

Several of the active learning approaches proposed in this study have successfully outperformed random sampling, while others have failed to do so. The successful results can be seen in Figure 4.1 and Figure 4.2. The unsuccessful techniques include active learning on the self-attention model, ACTS on the LSTM model, and GPLA. In the latter case, we speculate that this is due to the violation of the smoothness and cluster assumptions in the data.

One of our contributions in this study is the implementation of more novel methods for driver behavior classification. The novel methods are self-attention models and convolutional neural networks with joint recurrence plots. The benefits of using these models lie in their ability for parallel computation. Although simpler methods such as LSTM and 1D CNN performed better, they have some disadvantages to consider. LSTM models rely on sequential training, which is not feasible for very long time windows. The disadvantage of the 1D CNN is its restricted ability to capture very long-term temporal information, which is limited by the 1D CNN filter size.

Another contribution in this study is the implementation of novel active learning methods, which include ADD, GPLA, and ACTS. These techniques have shown

variable results across the experiments we have performed; the most successful techniques may provide a valid solution for improving the training process of classifiers. The benefit of using ADD is its efficiency in generating committee members for active learning approaches.

There are two limitations to this study that relate to data generation on the test track and the WICE dataset. The first limitation is that the test track data does not necessarily contain a real driving scenario, as considerably fewer drivers were used than most roads in the Gothenburg area during the day. This might imply that machine learning models trained on the test track data would not infer accurately on real-life driving data. The other limitation relates to the annotation techniques employed: since we have implemented an automatic annotation system based on the driving parameters proposed by Younes et al. [6], it may be more convenient to perform driving behavior classification through this system instead of machine learning classifiers. However, it may also be the case that performing inference on a trained model is more efficient than computing the driving parameters and the rules described in subsection 3.1.3 for each window of data. For a single window of 10 seconds, the annotation through the implemented system and the inference with the trained 1D CNN took approximately the same CPU time (0.05 seconds) on the same machine. Performing inference on a model with a lighter architecture is therefore likely to be more convenient than annotating with the rule-based system. Furthermore, the experiments on the test track data have shown that the implemented models can achieve good accuracy on data generated according to predefined driving styles.

A third limitation is the model's potential inability to generalize to a wider dataset. This limitation relates to both the WICE data and test track data. Both of these were gathered under similar road, weather, and traffic conditions. The conceptual framework from [7] mentions that all three of these factors affect driver behavior. Since our models are specifically trained on datasets where these factors are controlled, the models might not be able to generalize to a more general dataset.

5.3 Future work

Most works on driver behavior have performed analysis on data recreated using simulated driving behavior. This results in a dataset that is annotated as the data is gathered. However, many companies set up data pipelines that result in large pools of unlabeled datasets (e.g. WICE in our thesis). This results in a large dataset without any annotation on specific driving style. Clustering techniques have been performed in previous studies [8, 27, 29] that uncover structures in the data corresponding to different driving behaviors. There are also studies that implement different fuzzy controls [63] for computing driving behavior from driving signals.

However, our literature review revealed that there is a lack of research in trying to model driving behavior from manually labeled data. The manual labels would be produced by expertise in the domain of driving (i.e., export based examination

[64], eco-driving coaches [65], or using verification scores - see section A.1). There has also been research that focuses on labeling data using questionnaires and surveys [21, 23]. However, these questionnaires could be unreliable since they rely on the drivers' subjective experience [21]. Another interesting approach would be to manually label data from high-level features extracted from deep auto-encoders implemented by Liu et al. (2017) [18].

Future work may also focus on detecting aggressive driving in scenarios other than the ones covered during this project, e.g., within a city and over more trafficked roads. It may also be interesting to compare the correlation between aggressive driving and energy consumption in those scenarios, and incorporate the analysis in the forecasting of energy consumption based on driver behavior. Another interesting approach would be to adopt online learning to train the model continuously as driving data is generated.

Regarding active learning, we have seen a few examples of techniques that were generally less effective than random sampling for higher percentages of training data, but more effective in the early iterations of the active learning procedure, where the training data is scarce (e.g., active deep dropout on the 1D CNN). In the future it may be interesting to explore how these techniques fare with even smaller percentages for our training data, e.g. 1% or less.

Bibliography

[1] Peter Weldon, Patrick Morrissey, and Margaret O'Mahony. *"Environmental impacts of varying electric vehicle user behaviours and comparisons to internal combustion engine vehicle usage – An Irish case study"*. In: *Journal of Power Sources* 319 (2016), pp. 27–38. ISSN: 0378-7753. DOI: https://doi.org/10.1016/j.jpowsour.2016.04.051. URL: http://www.sciencedirect.com/science/article/pii/S0378775316304025.

[2] *McKinsey Electric Vehicle Index: Europe cushions a global plunge in EV sales*. July 2020. URL: https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/mckinsey-electric-vehicle-index-europe-cushions-a-global-plunge-in-ev-sales#.

[3] Johannes Kester. *"Security in transition(s): The low-level security politics of electric vehicle range anxiety"*. In: *Security Dialogue* 50.6 (2019), pp. 547–563. DOI: 10.1177/0967010619871443. URL: https://doi.org/10.1177/0967010619871443.

[4] *Electric car range: how far will they really go on a single charge?* July 2020. URL: https://www.buyacar.co.uk/cars/economical-cars/electric-cars/726/electric-car-range-how-far-will-they-really-go-on-a-single.

[5] C. Bingham, C. Walsh, and S. Carroll. *"Impact of driving characteristics on electric vehicle energy consumption and range"*. In: *IET Intelligent Transport Systems* 6.1 (2012), pp. 29–35. ISSN: 1751956X. DOI: 10.1049/iet-its.2010.0137.

[6] Zoulficar Younes et al. *"Analysis of the main factors influencing the energy consumption of electric vehicles"*. In: *Proceedings of the 2013 IEEE International Electric Machines and Drives Conference, IEMDC 2013* (2013), pp. 247–253. DOI: 10.1109/IEMDC.2013.6556260.

[7] Zouhair Elamrani Abou Elassad et al. *"The application of machine learning techniques for driving behavior analysis: A conceptual framework and a systematic literature review"*. In: *Engineering Applications of Artificial Intelligence* 87.August 2019 (2020), p. 103312. ISSN: 09521976. DOI: 10.1016/j.engappai.2019.103312. URL: https://doi.org/10.1016/j.engappai.2019.103312.

[8] Peng Ping et al. *"Impact of driver behavior on fuel consumption: Classification, evaluation and prediction using machine learning"*. In: *IEEE Access* 7 (2019), pp. 78515–78532. ISSN: 21693536. DOI: 10.1109/ACCESS.2019.2920489.

[9] Yang Xing et al. *"Energy oriented driving behavior analysis and personalized prediction of vehicle states with joint time series modeling"*. In: *Applied Energy* 261.December 2019 (2020), p. 114471. ISSN: 03062619. DOI: 10.1016/j.

apenergy.2019.114471. URL: https://doi.org/10.1016/j.apenergy.2019.114471.

[10] Mohammad Shahverdy et al. *"Driver behavior detection and classification using deep convolutional neural networks"*. In: *Expert Systems with Applications* 149 (2020), p. 113240. DOI: 10.1016/j.eswa.2020.113240. URL: https://doi.org/10.1016/j.eswa.2020.113240.

[11] Yoshihiro Nishiwaki et al. *"Driver Modeling Based on Driving Behavior and Its Evaluation in Driver Identification"*. In: *Proceedings of the IEEE* 95.2 (2007), pp. 1–11.

[12] Yoshihiro Nishiwaki et al. *"Generation of Pedal Operation Patterns of Individual Drivers in Car-Following for Personalized Cruise Control"*. In: *2007 IEEE Intelligent Vehicles Symposium*. 2007, pp. 823–827. DOI: 10.1109/IVS.2007.4290218.

[13] SangJo Choi et al. *"Analysis and classification of driver behavior using in-vehicle can-bus information"*. In: *Biennial Workshop on DSP for In-Vehicle and Mobile Systems* October 2015 (2007), pp. 17–19.

[14] N. Kuge et al. *"A Driver Behavior Recognition Method Based on a Driver Model Framework"*. In: *SAE transactions* 109 (2000), pp. 469–476.

[15] W. Takano et al. *"Recognition of human driving behaviors based on stochastic symbolization of time series signal"*. In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems* (2008), pp. 167–172.

[16] C. MacAdam et al. *"Using neural networks to identify driving style and headway control behavior of drivers"*. In: *Vehicle System Dynamics* 29.SUPPL. (1998), pp. 143–160. ISSN: 00423114. DOI: 10.1080/00423119808969557.

[17] Jun Zhang et al. *"A deep learning framework for driving behavior identification on in-vehicle CAN-BUS sensor data"*. In: *Sensors (Switzerland)* 19.6 (2019), pp. 6–8. ISSN: 14248220. DOI: 10.3390/s19061356.

[18] Hailong Liu et al. *"Visualization of Driving Behavior Based on Hidden Feature Extraction by Using Deep Learning"*. In: *IEEE Transactions on Intelligent Transportation Systems* 18.9 (Sept. 2017), pp. 2477–2489. ISSN: 15249050. DOI: 10.1109/TITS.2017.2649541.

[19] Cheng Zhang et al. *"Driver classification based on driving behaviors"*. In: *International Conference on Intelligent User Interfaces, Proceedings IUI*. Vol. 07-10-Marc. 2016, pp. 80–84. ISBN: 9781450341370. DOI: 10.1145/2856767.2856806.

[20] Jair Ferreira Júnior et al. *"Driver behavior profiling: An investigation with different smartphone sensors and machine learning"*. In: *PLoS ONE* 12.4 (2017), pp. 1–17. ISSN: 19326203. DOI: 10.1371/journal.pone.0174959. URL: http://dx.doi.org/10.1371/journal.pone.0174959.

[21] Jin-Hyuk Hong, Ben Margines, and Anind K. Dey. *"A Smartphone-Based Sensing Platform to Model Aggressive Driving Behaviors"*. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '14. Toronto, Ontario, Canada: Association for Computing Machinery, 2014, pp. 4047–4056. ISBN: 9781450324731. DOI: 10.1145/2556288.2557321. URL: https://doi.org/10.1145/2556288.2557321.

[22] Motonori Ishibashi et al. *"Indices for characterizing driving style and their relevance to car following behavior"*. In: *SICE Annual Conference 2007*. 2007, pp. 1132–1137. DOI: 10.1109/SICE.2007.4421155.

[23] Rishu Chhabra, C. Rama Krishna, and Seema Verma. *"Smartphone based context-Aware driver behavior classification using dynamic Bayesian network"*. In: *Journal of Intelligent and Fuzzy Systems* 36.5 (2019), pp. 4399–4412. ISSN: 18758967. DOI: 10.3233/JIFS-169995.

[24] Chaoyun Zhang et al. *"Driver behavior recognition via interwoven deep convolutional neural nets with multi-stream inputs"*. In: *arXiv* 8 (2018). ISSN: 23318422. DOI: 10.1109/access.2020.3032344. arXiv: 1811.09128.

[25] K. Ihme et al. *"Recognizing Frustration of Drivers From Face Video Recordings and Brain Activation Measurements With Functional Near-Infrared Spectroscopy"*. In: *Frontiers in Human Neuroscience* 12 (2018).

[26] Fred Feng et al. *"Can vehicle longitudinal jerk be used to identify aggressive drivers? An examination using naturalistic driving data"*. In: *Accident Analysis and Prevention* 104.May (2017), pp. 125–136. ISSN: 00014575. DOI: 10.1016/j.aap.2017.04.012. URL: http://dx.doi.org/10.1016/j.aap.2017.04.012.

[27] Z. Constantinescu, C. Marinoiu, and M. Vladoiu. *"Driving style analysis using data mining techniques"*. In: *International Journal of Computers, Communications and Control* 5.5 (2010), pp. 654–663. ISSN: 18419844. DOI: 10.15837/ijccc.2010.5.2221.

[28] Xin Lin et al. *"Driver Evaluation and Identification Based on Driving Behavior Data"*. In: *Proceedings - 2018 5th International Conference on Information Science and Control Engineering, ICISCE 2018* (2019), pp. 718–722. DOI: 10.1109/ICISCE.2018.00154.

[29] Zahid Halim, Rizwana Kalsoom, and Abdul Rauf Baig. *"Profiling drivers based on driver dependent vehicle driving features"*. In: *Applied Intelligence* 44.3 (2016), pp. 645–664. ISSN: 15737497. DOI: 10.1007/s10489-015-0722-6.

[30] David D. Lewis and William A. Gale. *"A Sequential Algorithm for Training Text Classifiers"*. In: *CoRR* abs/cmp-lg/9407020 (1994). arXiv: cmp-lg/9407020. URL: http://arxiv.org/abs/cmp-lg/9407020.

[31] Gokhan Tur, Dilek Hakkani-Tur, and Robert Schapire. *"Combining active and semi-supervised learning for spoken language understanding"*. In: *Speech Communication* 45 (Feb. 2005), pp. 171–186. DOI: 10.1016/j.specom.2004.08.002.

[32] Ying Liu. *"Active Learning with Support Vector Machine Applied to Gene Expression Data for Cancer Classification"*. In: *Journal of Chemical Information and Computer Sciences* 44.6 (2004). PMID: 15554662, pp. 1936–1941. DOI: 10.1021/ci049810a. eprint: https://doi.org/10.1021/ci049810a. URL: https://doi.org/10.1021/ci049810a.

[33] Simon Tong. *"Active Learning: Theory and Applications"*. PhD thesis. Stanford University, 2001. URL: http://www.robotics.stanford.edu/~stong/papers/tong_thesis.pdf.

[34] Burr Settles. *Active Learning Literature Survey*. Computer Sciences Technical Report 1648. University of Wisconsin–Madison, 2009.

[35] Martin Gammelsæter. *"A Committee of One"*. MA thesis. Norwegian University of Science and Technology, Aug. 2015. URL: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/2352342.

[36] John Daniel Bossér, Erik Sörstadius, and Morteza Haghir Chehreghani. *"Model-centric and data-centric aspects of active learning for neural network models"*. In: *arXiv* (2020). ISSN: 23318422. arXiv: 2009.10835. URL: http://arxiv.org/abs/2009.10835.

[37] Fengchao Peng, Qiong Luo, and Lionel M. Ni. *"ACTS: An active learning method for time series classification"*. In: *Proceedings - International Conference on Data Engineering* 0 (2017), pp. 175–178. ISSN: 10844627. DOI: 10.1109/ICDE.2017.68.

[38] Ashish Vaswani et al. *"Attention is all you need"*. In: *Advances in Neural Information Processing Systems* 2017-Decem.Nips (2017), pp. 5999–6009. ISSN: 10495258. arXiv: arXiv:1706.03762v5.

[39] Saif Mahmud et al. *"Human activity recognition fromwearable sensor data using self-attention"*. In: *Frontiers in Artificial Intelligence and Applications* 325 (2020), pp. 1332–1339. ISSN: 09226389. DOI: 10.3233/FAIA200236. arXiv: arXiv:2003.09018v1.

[40] Nikiforos Zacharof et al. *"Review of in use factors affecting the fuel consumption and CO2 emissions of passenger cars"*. In: (Jan. 2016). DOI: 10.2790/140640.

[41] *UTDrive - Research Platform for In-Vehicle Safety Systems and Driver Behavior Modeling*. URL: https://www.utdallas.edu/research/utdrive/index.html.

[42] Béatrice Cahour et al. *"Using an Electric Car: A Situated, Instrumented and Emotional Activity"*. In: *Proceedings of the 30th European Conference on Cognitive Ergonomics*. ECCE '12. Edinburgh, United Kingdom: Association for Computing Machinery, 2012, pp. 22–28. ISBN: 9781450317863. DOI: 10.1145/2448136.2448142. URL: https://doi.org/10.1145/2448136.2448142.

[43] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[44] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: a modern approach*. 3rd ed. Pearson, 2009.

[45] *Convolution - Wolfram MathWorld*. URL: https://mathworld.wolfram.com/Convolution.html.

[46] Yann LeCun et al. *"Object Recognition with Gradient-Based Learning"*. In: *Shape, Contour and Grouping in Computer Vision*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 319–345. ISBN: 978-3-540-46805-9. DOI: 10.1007/3-540-46805-6_19. URL: https://doi.org/10.1007/3-540-46805-6_19.

[47] Sepp Hochreiter and Jürgen Schmidhuber. *"Long Short-term Memory"*. In: *Neural computation* 9 (Dec. 1997), pp. 1735–80. DOI: 10.1162/neco.1997.9.8.1735.

[48] Felix Gers, Jürgen Schmidhuber, and Fred Cummins. *"Learning to Forget: Continual Prediction with LSTM"*. In: *Neural computation* 12 (Oct. 2000), pp. 2451–71. DOI: 10.1162/089976600300015015.

[49] Laurent Itti, Christof Koch, and Ernst Niebur. *"A Model of Saliency-based Visual Attention for Rapid Scene Analysis"*. In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 20 (Dec. 1998), pp. 1254–1259. DOI: 10.1109/34.730558.

[50] Stanislaw Weglarczyk. *"Kernel density estimation and its application"*. In: *ITM Web of Conferences* 23 (Jan. 2018), p. 00037. DOI: 10.1051/itmconf/20182300037.

[51] Marina Sokolova and Guy Lapalme. *"A systematic analysis of performance measures for classification tasks"*. In: *Information Processing Management* 45 (July 2009), pp. 427–437. DOI: 10.1016/j.ipm.2009.03.002.

[52] *Keras - accuracy metrics*. URL: https://keras.io/api/metrics/accuracy_metrics/#categoricalaccuracy-class.

[53] Tom Fawcett. *"Introduction to ROC analysis"*. In: *Pattern Recognition Letters* 27 (June 2006), pp. 861–874. DOI: 10.1016/j.patrec.2005.10.010.

[54] *UEvaluation Metrics - RDD-based API*. URL: https://spark.apache.org/docs/latest/mllib-evaluation-metrics.html.

[55] *Google Maps: Testbana, 418 78 Göteborg*. Maps Data: Google, ©2021 Aerodata International Surveys, CNES / Airbus, Landsat / Copernicus, Lantmäteriet / Metria, Maxar Technologies. URL: https://goo.gl/maps/P4bSQLDJdmCqM8HY7.

[56] Heikki Summala. *"Brake Reaction Times and Driver Behavior Analysis"*. In: *Transportation Human Factors* 2.3 (2000), pp. 217–226. ISSN: 1093-9741. DOI: 10.1207/sthf0203_2.

[57] Daniel V. McGehee, Elizabeth N. Mazzae, and G. H.Scott Baldwin. *"Driver reaction time in crash avoidance research: Validation of a driving simulator study on a test track"*. In: *Proceedings of the XIVth Triennial Congress of the International Ergonomics Association and 44th Annual Meeting of the Human Factors and Ergonomics Association, 'Ergonomics for the New Millennium'* May 2014 (2000), pp. 320–323. ISSN: 1541-9312. DOI: 10.1177/154193120004402026.

[58] Jiadi Yu et al. *"Fine-Grained Abnormal Driving Behaviors Detection and Identification with Smartphones"*. In: *IEEE Transactions on Mobile Computing* 16.8 (2017), pp. 2198–2212. DOI: 10.1109/TMC.2016.2618873.

[59] Andrei Aksjonov et al. *"A Novel Driver Performance Model Based on Machine Learning"*. In: *IFAC-PapersOnLine* 51.9 (2018). 15th IFAC Symposium on Control in Transportation Systems CTS 2018, pp. 267–272. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2018.07.044. URL: https://www.sciencedirect.com/science/article/pii/S2405896318307675.

[60] Fabio Tango and Marco Botta. *"Real-Time Detection System of Driver Distraction Using Machine Learning"*. In: *IEEE Transactions on Intelligent Transportation Systems* 14.2 (2013), pp. 894–905. DOI: 10.1109/TITS.2013.2247760.

[61] Olivier Chapelle, Bernhard Schölkopf, and Alexander Zien. *Semi-Supervised Learning*. Adaptive computation and machine learning. MIT Press, 2006. ISBN: 0262033585,978-0-262-03358-9.

[62] David Lowell, Zachary Lipton, and Byron Wallace. *"Practical Obstacles to Deploying Active Learning"*. In: Jan. 2019, pp. 21–30. DOI: 10.18653/v1/D19-1003.

[63] Na Lin et al. *"An overview on study of identification of driver behavior characteristics for automotive control"*. In: *Mathematical Problems in Engineering* 2014.2 (2014). ISSN: 15635147. DOI: 10.1155/2014/569109.

[64] Sami Mynttinen et al. *"Self-assessed driver competence among novice drivers – a comparison of driving test candidate assessments and examiner assessments in a Dutch and Finnish sample"*. In: *Journal of Safety Research* 40.4 (2009), pp. 301–309. ISSN: 0022-4375. DOI: https://doi.org/10.1016/j.jsr.2009.04.006. URL: https://www.sciencedirect.com/science/article/pii/S0022437509000747.

[65] CGI. *"Modeling the Relation Between Driving Behavior and Fuel Consumption"*. In: (2014), pp. 1–10.

A

Appendix 1

A.1 Annotation techniques for driver behavior

A.1.1 Aggressivity index

One approach for annotation proposed by MacAdams et al. [16] is to compare drivers by *aggressivity index*. Let AI be the aggressivity index ranging from 0 to 1. This index can be described as the willingness of a driver to overtake other vehicles. It can be computed using the following percentages: *closing-in-rapidly* (CIR), *closing-in* (CI), and *following* (F). The AI can be computed in the following manner:

$$AI = (CIR + CI) + \frac{F}{2} \tag{A.1}$$

A high AI score would describe a driver who passes all other vehicles and never follows nor is passed by other vehicles. An AI score of 0.5 would describe a driver who only spends time following other vehicles. A low AI score would describe a driver who lets all other drivers overtake them.

A.1.2 Fuzzy rules

Another approach for annotating driver behavior is using Table A.1. This table was designed by Lin et al. (2014). The "fuzziness" refers to the fact that these rules are not set numerically. [63].

Gap time	Accelerator pedal rate STD	Brake pedal rate STD	Driving style
Low	Low	Low	Less Aggressive
High	Low	Low	Cautious
Low	High	Low	Aggressive
Low	Low	High	Aggressive
Low	High	High	Aggressive
High	High	High	Less Aggressive
High	Low	High	Cautious
High	High	Low	Less Aggressive

Table A.1: Fuzzy rules explained in [63].