The impact of parsing methods on recurrent neural networks applied to event-based vehicular signal data

MAX EVERT LINDBLAD
The impact of parsing methods on recurrent neural networks applied to event-based vehicular signal data

Påverkan av parsningsmetoder på återkommande neuronnät applicerade på händelsebaserad signaldata från fordon

Author: Max Lindblad - Email: maxlindb@kth.se
Department: Computer Science - Employer: Scania AB
Examiner: Joakim Gustafson - Supervisor: Pawel Herman
Date: 2018-03-07
Abstract

This thesis examines two different approaches to parsing event-based vehicular signal data to produce input to a neural network prediction model: event parsing, where the data is kept unevenly spaced over the temporal domain, and slice parsing, where the data is made to be evenly spaced over the temporal domain instead. The dataset used as a basis for these experiments consists of a number of vehicular signal logs taken at Scania AB. Comparisons between the parsing methods have been made by first training long short-term memory (LSTM) recurrent neural networks (RNN) on each of the parsed datasets and then measuring the output error and resource costs of each such model after having validated them on a number of shared validation sets. The results from these tests clearly show that slice parsing compares favourably to event parsing.

Sammanfattning

Denna avhandling jämför två olika tillvägagångssätt vad gäller parsningen av händelsebaserad signaldata från fordon för att producera indata till en förutsägelsemodell i form av ett neuronett, nämligen händelseparsning, där datan förblir ojämnt fördelad över tidsdomänen, och skivparsning, där datan är omgjord till att istället vara jämnt fördelad över tidsdomänen. Det dataset som används för dessa experiment är ett antal signalloggar från fordon som kommer från Scania. Jämförelser mellan parsningsmetoderna gjordes genom att först träna ett lång korttidsminne (LSTM) återkommande neuronnett (RNN) på vardera av de skapade dataseten för att sedan mäta utmatningsfelet och resurskostnader för varje modell efter att de validerats på en delad uppsättning av valideringsdata. Resultaten från dessa tester visar tydligt på att skivparsning står sig väl mot händelseparsning.
Chapter 1

Introduction

The automotive industry is a treasure trove for machine learning problems, where the possibilities are everything from customer analytics to the creation of self-driving cars. Unsurprisingly, a considerable amount of research has already been done on machine learning applications in relation to the automotive industry [1,2]. However, there still exist many types of datasets within the sector for which the potential of using machine learning methods has not yet been properly explored, even though this could be greatly beneficial. One such dataset is the vehicular signal data available at the vehicle manufacturer Scania AB. Scania AB has long had a system which allows users to customize their truck purchases. In essence, the user is allowed to choose from a number of different modules, which together make up a complete configuration of a vehicle. However, there is no guarantee that a combination will work properly without having tested it first (e.g. perhaps a certain combination of modules causes the brake light to stop working). As having to physically construct and drive every possible configuration in a real-world scenario is infeasible, Scania AB also checks if everything conforms to quality standards by testing these vehicles in more controlled environments, which includes simulations. Each signal sent during these simulations wherein a property, such as the vehicle speed, is changed is then recorded into a log for that simulation, which together make up the aforementioned dataset.

The extraction of temporal patterns from these logs could be used for a number of different applications. Of these applications, the extraction of domain knowledge from the data is currently the most desired by Scania AB since determining how the different signals relate would greatly contribute to their existing projects on the subject of improving testing efficiency [3]. A potential way to gain this information would be to train a prediction model on the data. A model that can accurately predict multisensory array signals should have knowledge about how the individual signals interact with one another [4,5]. Although hidden Markov models and recurrent neural networks (RNNs) have both been commonly used as a basis for prediction models in the past [4,6], neural network based approaches have in recent years proven to be better suited for handling high-dimensional data than hidden Markov model based approaches [7]. As such, without any additional information, RNNs are currently the most attractive avenue to pursue for accomplishing Scania’s goals with regards to the vehicular signal data.

A considerable challenge with using the vehicular signal data is however its event-based nature, reflected in the fact that the samples are unevenly spaced over the temporal domain, as this means that it cannot be parsed in the same way as evenly spaced data. This is problematic because there is little research and a general lack of information on how to parse sequences of irregular multivariate events into a representation suitable for most machine learning methods, including RNNs. The few papers that touch upon this subject, e.g. [8], do
not address the problem in the general case. Although there currently does not exist a clearly defined set of methods for parsing event-based data, it is possible to categorise the state-of-the-art approaches into two groups: methods that parse event-based data as is in the form of unevenly spaced data [9], and methods that instead convert the event-based data into slice-based data, in which the original events are parsed into evenly spaced time-slices [10]. Still, no comparisons have been made between how the different parsing methods affect the performance of the resulting machine learning models in terms of output error and associated resource costs (i.e. computational and memory costs).

1.1 Problem statement
The crux of the problem with designing a neural network based system for processing event-based vehicular signal data is the choice of an efficient representation. A quantitative comparative study of different methods for parsing the vehicular signal data into a prediction model is expected to provide valuable insights into this issue, and pave the way for future work on how to efficiently use RNNs with event-based data. The benefits of that study would not be limited to just the specific context of applying RNNs to vehicular signal data either, as there is a severe lack of information on the subject of applying machine learning methods to event-based data in general.

The research question to be addressed in this thesis can be formulated as follows:
Which of the two proposed parsing methods is more suitable for preprocessing event-based vehicular signals for an RNN prediction model? The criteria of interest encompass the overall prediction accuracy and resource costs (i.e. computational and memory costs) of the model.

1.2 Thesis objective and scope
The aim of this thesis is to evaluate the effect that different types of parsing methods for event-based data have on prediction models applied to vehicular signal data, i.e. RNNs trained to predict future signals from prior signals, with respect to their prediction accuracy as well as their resource costs. However, the development of entirely new types of parsing methods and/or neural network algorithms is beyond the scope of this project. Instead, this paper only serves to evaluate the two most common parsing methods for event-based data.

1.3 Thesis outline
The thesis consists of five chapters in the following order:
Chapter 2: Background, where a more detailed overview of the scientific context for this thesis is presented.
Chapter 3: Methods, where the precise methods used for the experiments are detailed and explained.
Chapter 4: Results, where the specific experiments performed are described and their results shown.
Chapter 5: Discussion, where the implications of these results are discussed.
Chapter 6: Conclusions and future work, where a brief summary of the results as well as possible future work is presented.
Chapter 2

Background

2.1 Neural networks

Multi-layered artificial neural networks (see an example of a two-hidden-layer architecture in Figure 1) are a group of machine learning models that rely on a connectionist information processing approach and that through learning can extract useful data representations without the same reliance on hand-engineered features as the more traditional statistical methods, though the model configuration itself still needs to be tuned for each task [11,12]. These networks consist of a multitude of small and inexpensive artificial neurons that are separated into layers. While an individual neuron has a very limited amount of computational power, combining several together results in a model that is capable of both learning and expressing complex relationships between variables. Each layer in the network represents another layer of abstraction. Neural networks’ greatest strength is that they make relatively few assumptions about the patterns in the data. On the other hand, the type of training that involves non-convex optimisation poses a severe risk of overfitting especially given limited training data. Neural networks therefore tend to require larger datasets and longer training times than other types of machine learning models [13].

Figure 1: Visual representation of a neural network with two hidden layers, where the input is a set of six variables, and the target output is a singular variable. The output of each neuron (unit) in a previous layer is weighted and provided as an input for each neuron (unit) in the next layer (this process being represented by the arrows).
2.1.1 Recurrent Neural Networks

By default, neural networks do not have any memory, and are thus ill-suited for the modelling of temporal patterns in sequential data. As a result, an extension called recurrent neural networks (RNN) was introduced for the regular neural networks (shown in Figure 2) [14]. Recurrent neural networks differ from regular neural networks in that they have at least one feedback loop, wherein the output of a hidden layer is not just dependent on its current input, but also its previous output. Thus, for recurrent neural networks, the hidden layers also serve as a form of memory. Originally, training these networks to recognize long-term temporal patterns was a very difficult task, as the gradient that was meant to update the weights in the network would eventually either vanish or explode after it had gone through too many temporal steps [15].

Long short-term memory (LSTM) networks within the family of RNNs were presented as a remedy for this problem [16], which avoid the problem of exploding / vanishing gradients by using a linear activation function between the temporal steps, Thereby allowing neural networks to be used to learn temporal patterns that span over much longer time periods than before. LSTMs have since shown to be a capable tool for the purpose of extracting both short-term and long-term temporal patterns from sequential data, and have been widely applied to high-profile problems such as speech recognition and video recognition [11]. In those applications, LSTMs are either used as a stand-alone model (i.e. an end-to-end neural model) [17], or in combination with other machine learning methods (i.e. a hybrid model) [18]. While hybrid models have traditionally outperformed end-to-end models, this gap has shrunk over the years and end-to-end models are now managing to achieve state-of-the-art
performance in some fields [18]. Furthermore, end-to-end systems have the additional advantage over hybrid models in that even less human effort and/or prior knowledge is required for their application [10], and are therefore well suited for application on more novel datasets. However, choosing an appropriate output-target and error function is critical even for end-to-end models, and much of the recent research has been concerned with the development of better algorithms for this purpose, with the prime example of the connectionist temporal classification (CTC) method designed for speech recognition [33].

Although RNNs, like most types of neural networks, have primarily been applied to supervised problems [11], a great deal of work has also been done on the subject of training RNNs in an unsupervised fashion, as for most supervised problems there will be considerably less labeled data than unlabeled data available [12]. In recent literature one can find two different techniques for the unsupervised training of RNNs. Training the RNN to reconstruct its input data (i.e. an autoencoder model), and/or training the RNN to predict the next data point of a truncated sequence (i.e. a future predictor model) [4,19]. However, as it is always possible for the autoencoder model to get a lower output error by simply copying its inputs, using a future predictor model is preferable if the goal is to compare how well two models are doing [4].

2.2 Parsing event-based data

Datasets consisting of event sequences, i.e. unevenly spaced time series data in the format $S_e = \{(e_1, t_1), (e_2, t_2), \ldots, (e_m, t_m)\}$ where $e$ is an event and $t$ is the timestamp for that event, have been found in a number of different disciplines, examples of datasets would be medical records, activity diary data, and internet surfing history [9,20]. In addition, there are two special cases of evenly spaced time series that are naturally treated as being unevenly spaced. These are time series with missing observations, and multivariate data sets that consist of time series with different frequencies [21]. However, as many of the traditional methods for analysing time series data are not valid for event-based data, the most common approach for handling event-based data has been to first transform it into slice-based data [21,22]. Wherein the original sequences are converted into the format $S_s = \{s_1, s_2, \ldots, s_k\}$ where $s$ is a temporal slice containing the events for a specific time period.

The transformation of event-based data into slice-based data has predominantly been done via interpolation, primarily linear interpolation, due to the frequency of event-based datasets being caused by missing observations [21]. By interpolating, the gaps that contain no observations / events can be filled in, and the temporal spacing between each frame standardized [21]. In some cases the data representation is also changed as a part of the interpolation process, in which measures of speed are for example replaced by measures of acceleration [22]. Due to overly high rate of sampling for many types of event-based data, the
data will also commonly be subjected to downsampling [23]. As with the conversion of event-based data into slice-based data, interpolation techniques are often used for this purpose, although with decimation as an alternative [23].

However, heavily changing the representation of event sequences or time series data in general is not without drawbacks, and modifying the data to make it easier to use can result in it becoming severely distorted, to the point where important patterns are lost [20,21]. Avoiding this loss of information has been the main drive behind the development of improved methods for interpolation and downsampling [24]. It is also the reason why there exists a considerable amount of research on the subject of investigating methods that can be directly applied to event-based data [9,21]. As the use of such methods precludes the need to change the representation of the data.

### 2.3 Automotive industry

The basis for this thesis is the research project called Situation-Based Integration Testing of Automotive Systems using Guarded Assertions (SAGA). It is a cooperation project between Scania and SICS (Swedish Institute of Computer Science) with the goal of automating the testing process for the vehicles at Scania AB [3]. Initial work on the project solely focused on developing methods for continuous evaluation during testing, but now the research project also includes the investigation of ways to improve the accuracy of the test stimulus. Currently, their goal is to train an RNN on the vehicular signal data to have domain knowledge about the underlying patterns.

Beyond the work at Scania AB, this thesis also has a clear connection to the more general domain of car diagnostics and other types of vehicle monitoring. Most research within this field has revolved around the development of systems that serve to simplify the manual diagnosis/monitoring of automobiles [25,26], rather than to completely automate the process. These manual systems commonly utilize several different data sources, occasionally including signal data, to provide a comprehensive overview of the current situation to the user. Although these applications have generally been successful, there would be obvious advantages to replacing them with ones that are fully autonomous.

### 2.4 Related work

While there is a large amount of research available regarding both the application of RNNs to time series data (e.g. [4]), and the use of event-based data for analysis (e.g. [27]), there exist remarkably few papers within the crossover region between these two fields. In addition, no references could be found in literature on the subject of using event-based data for analytical tasks within the automotive industry. For event-based data in general, research has been primarily concerned with solving conventional data mining tasks such as the extraction of interesting sequences or similarity calculations between one or more event sequences via the use of more traditional numerical methods, i.e. not machine learning methods [9]. Instances
where any type of neural network has been mentioned in this circumstance are, as such, exceedingly rare, with the choice of representation having been done in an ad-hoc manner in the few examples that exist [8,10].

These chosen representations for the event-based data do, however, generally involve converting it into slice-based data, and then using some form of downsampling. As in [8], where a convolutional neural network was used for stock market prediction, the representation for the events was a series of spatio-temporal slices, each slice corresponding to a specific level of temporal granularity. While for [10], where a multilayer perceptron was used to predict online user purchases, the temporal element of the events was marginalized completely. Moreover, though it has been claimed that applying machine learning algorithms to event-based data is not fundamentally different from applying them to other types of time-series data [28], there are no examples of RNNs being applied to event-based data directly.

In any case, machine learning models trained on multivariate time series data have been shown to often suffer from overfitting and poor generalization, especially in the case of real-world multivariate time series data (such as vehicular signal data) due to the prevalence of irrelevant and/or superfluous variables in those types of datasets [29]. It is therefore important to perform variable selection when dealing with high-dimensional time series data in this context [30]. Variable selection techniques are varied [31], and can range from simply hand-picking variables [31], to independent component analysis (ICA) [32] and even more complicated methods. Which method is used depends largely on the dataset at hand, with high amounts of domain knowledge allowing for simpler variable selection schemes [31].
Chapter 3

Methods

3.1 Data
The vehicular signal data available to this study consisted of a total of 94 controller area network (CAN) bus logs provided by Scania AB. Each log detailed when – in the form of timestamps – and how – in the form of new values – each of the tracked variables, 4518 in total, changed when the test suite was subjected to a specific set of test-stimuli. On average a new value was assigned to one of the variables every 0.009 seconds, with no singular variable changing more often than once every 0.01 seconds. The type of value that these variables represented varied greatly regarding everything from the current date to the speed of the vehicle. Thus, because of the large number of disparate variables, the variables were screened to conserve memory and facilitate efficient learning.

Initial screening limited the variables to those that had at some point been tracked in the manually created test cases at Scania AB, as those were deemed to have a higher chance of containing relevant information. Thereafter, the variables that had very rarely been changed in the available CAN-logs were removed from the data pool since their values would only be taken into account if the model was overfitting. Lastly, the nature of the values was also considered to a certain degree, for example any information regarding the date that the test occurred was ignored as it would have been irrelevant for predicting the next frame. The end result was a total of 45 continuous variables (e.g. measures of vehicle speed, cabin temperature, pedal position, brake pressure, engine torque, and so on) extracted from the original 4518.

Without further processing, the baseline representation for the data in each log was therefore a sequence with the format,
\[ S = \{(\bar{e}_1, t_1), (\bar{e}_2, t_2), \ldots, (\bar{e}_m, t_m)\} \]
where each \( \bar{e} \) specifies a new value for one of the 45 variables by a non-zero value for that variable in a 45-dimensional vector, and the respective \( t \) is the timestamp for that change.

3.2 Parsing
The following subsections describe how the parsing methods used in the experiments were constructed, and in Figure 3 their differences are visualized.

3.2.1 Event parsing
The first parsing method – event parsing – revolved around largely keeping the signal data in its original format, i.e. as unevenly spaced event sequences. Only two modifications were therefore made to the original signal data when parsing it into these event sequences.
Firstly, the relatively few events that shared a timestamp were grouped together to make the data less sparse, as they could safely be represented by a single event. Secondly, events were made to be represented by the value of its changes, rather than by the end result of those changes. As it would otherwise not have been possible to compare the predictions made by each model in cases where they worked at different time scales. An alternative solution to this problem was considered in which the current values for every variable were included in each event. But this solution was eventually rejected, as it could have caused the network to primarily learn trivial correlations rather than the sought-after more complex correlations as each frame would then only slightly differ from the previous frame.

The mathematical expression for how the original event sequences, \( S = \{(\bar{e}_1, t_1), (\bar{e}_2, t_2), \ldots, (\bar{e}_m, t_m)\} \)

were converted into the new, parsed, event sequences \( S_e \); is thus as follows,

\[
S_e = \left\{ \sum_{i=1}^{m} s_i, t'_n \right\} \text{ for all distinct } t_n \in S
\]

for which (given \( t_n \))

\[
s_i = \begin{cases} 
\bar{e}_i', & \text{if } t_i = t_n \\
0, & \text{otherwise}
\end{cases}
\]

where

\[
\bar{e}_i' = \bar{Q}_i - \bar{Q}_{i-1},
\]

\[
t'_i = t_i - t_{i-1},
\]

and

\[
\bar{Q}_{i,r} = \begin{cases} 
\bar{e}_{i,r}, & \text{if } \bar{e}_{i,r} \neq 0 \\
\bar{Q}_{i-1,r}, & \text{otherwise}
\end{cases}
\]

for each variable \( r \) and event index \( i \). Note that these new sequences will have the format

\[
S_e = \{(\bar{e}_1', t'_1), (\bar{e}_2', t'_2), \ldots, (\bar{e}_l', t'_l)\}
\]

which is almost identical to the original format

\[
S = \{(\bar{e}_1, t_1), (\bar{e}_2, t_2), \ldots, (\bar{e}_m, t_m)\}
\]

except for the two previously described differences, i.e. that the relative measures \( \bar{e}' \) and \( t' \) are used rather than absolute measures \( \bar{e} \) and \( t \), and that events that share a timestamp are grouped together (thus making the sequence shorter).

### 3.2.1.1 Rounded event parsing

A variation of the event parsing method was also explored, which attempted to utilize the fact that no singular variable would change more often than once every 0.01 seconds by reducing the temporal resolution of the original events, i.e. the timestamps in the original data were rounded to a precision 0.01 seconds. Moreover, although this was never tested, it would have been possible to reduce the temporal resolution of the original events even further, at the risk of losing information about some of the changes that occurred.
Thus, rounded event parsing differs from regular event parsing in that there is an extra step, wherein each timestamp $t$ in the original event sequences

$$S = \{(\vec{e}_1, t_1), (\vec{e}_2, t_2), \ldots, (\vec{e}_m, t_m)\}$$

is assigned a new value according to the equation,

$$t = \frac{[10^{R_f}]}{10^R}$$

where $R$ is the new level of precision for the timestamps (specified by the desired number of decimals), before the event parsing method was applied.

### 3.2.2 Slice parsing

For the other parsing method – slice parsing – the objective was to convert the unevenly spaced event sequences into evenly spaced sequences containing spatio-temporal slices. While it is common to use interpolation for this purpose, it was unnecessary to do so in this case because no observations were missing from the data. Instead, all variables could be assumed to be constant until an event occurred, and it was thus sufficient to reduce the time resolution of the original events to desired precision (determined by the slice spacing) and then fill out the empty sections with zero-set time-slices.

The mathematical expression for how the original event sequences,

$$S = \{(\vec{e}_1, t_1), (\vec{e}_2, t_2), \ldots, (\vec{e}_m, t_m)\}$$

were converted into the new, parsed, slice sequences $S_s$; is thus as follows,

$$S_s = \left\{ \sum_{i=1}^{m} s_i \text{ for } n = 1, 2, \ldots, \left\lfloor \frac{t_m}{L} \right\rfloor \right\}$$

for which (given $n$)

$$s_i = \begin{cases} \vec{e}_i' & \text{if } Ln > t_i \geq L(n - 1) \\ 0 & \text{otherwise} \end{cases}$$

where $L$ is the slice spacing and $\vec{e}_i'$ is, again, the relative measure for the respective change.

Unlike the event parsing method, however, these parsed sequences do not share the original format, and instead have the format,

$$S_s = \left\{ s_1', s_2', \ldots, s_k' \right\}$$

where $s'$ is a time-slice containing relative measures for all the changes that occurred within the respective time period. Make notice that since the temporal spacing $t_i'$ between two adjacent slices $s_i'$ and $s_{i-1}'$ will always be equal to the slice spacing $L$, this information was not included in the slice sequences.

Furthermore, because of the magnitude of the temporal spacing between some of the original events, the creation of slices for every possible timespan had the risk of leading to an excessive amount of zero-set time-slices. Therefore, when the time between two events
exceeded a specified limit (the cut-off limit), the current slice sequence was ended and a new one began.

Figure 3: Visual representation of the differences between the parsing methods via a fictional event sequence covering three variables, X, Y, Z. The first row, i.e. no parsing, shows the original unparsed format (section 3.1) for that event sequence, while the sequences in the following rows each correspond to one of the previously described parsing methods. These are; event parsing (section 3.2.1), event parsing with rounding (section 3.2.1.1), and slice parsing (section 3.2.2). Note that in this fictional example the timestamps were rounded to a precision of 0.1 seconds for the event parsing method that used rounding, and not to a precision of 0.01 seconds as was done in the actual implementation.

3.2.3 Normalization
The data was normalized by extracting the standard score for each value in each frame, with the mean and standard deviation having been calculated from the available data. This process was done separately for each sequence set, i.e. no unified mean or standard deviation existed between them. For the event sequences, the temporal variable $t'$ was also normalized, as the value of $t'$ for an event will be dependent on the value of the non-temporal variables $\vec{v}$ and vice versa.

Therefore, each parsed event sequence

$$S_e = \{(\vec{v}_1', t_1'), (\vec{v}_2', t_2'), \ldots, (\vec{v}_l', t_l')\}$$
was normalized into the sequence $N_e$ via the following process using the means $\mu_1$ and $\mu_t$ and the standard deviations $\sigma_1$ and $\sigma_t$ for that specific dataset (with $\odot$ denoting element-wise division):

$$N_e = ((e'_1 - \mu_e) \odot \sigma_e, \frac{t'_1 - \mu_t}{\sigma_t}), ((e'_2 - \mu_e) \odot \sigma_e, \frac{t'_2 - \mu_t}{\sigma_t}), \ldots, ((e'_t - \mu_e) \odot \sigma_e, \frac{t'_t - \mu_t}{\sigma_t})$$

while each parsed slice sequence $S_s = \{ s'_1, s'_2, \ldots, s'_k \}$

was normalized into the sequence $N_s$ via the following, equivalent, process using the mean and standard deviation $\mu_s$ and $\sigma_s$ for that specific dataset,

$$N_s = \{(s'_1 - \mu_s) \odot \sigma_s, (s'_2 - \mu_s) \odot \sigma_s, \ldots, (s'_k - \mu_s) \odot \sigma_s \}$$

### 3.3 Model
An Encoder-Decoder LSTM neural network, as described in [4] and shown in Figure 4, served as the prediction model in all experiments. LSTMs have in general proven to be quite capable of extracting temporal patterns from sequential data [11], and the LSTM Encoder-Decoder architecture is a stable and effective variant that can be applied to many different kinds of sequence-to-sequence learning tasks [4, 21, 34]. Because the goal was to compare parsing methods and not different model configurations, it was decided that the same parameters and setup would be used for each model. Consequently, the only point of variation between models was the format used for their input and output sequences.

Each Encoder-Decoder model was divided into two parts; the encoder, which encoded the variable-length input into a fixed length representation, and the decoder, which then converted (decoded) this representation into the actual output for the model. Both the encoder and the decoder consisted of an LSTM neural network, and the shared model configuration used the same number of layers and hidden units in both the encoder and decoder, with the addition of a single, linear output layer at the end of each LSTM cell in the decoder. For the models that used an event-based representation, this linear output layer also placed a non-negative constraint on the temporal variable in order to ensure that it would be positive. Allowing for negative values would have risked causing problems when attempting to summarize the predictions of an event-based model over a time period. Other parameters such as the exact number of layers and hidden units were chosen based on a tradeoff between training time and performance for the most resource-intensive model.

To improve prediction quality, the choice was also made to have the decoder be conditioned, in which the output from each LSTM cell in the final layer of the decoder is added to input of the next LSTM cell in the first layer of the decoder, rather than unconditioned. Conditioning does however risk causing the model to focus on learning trivial, short-range correlations, since the previous output will then directly influence the next output. Fortunately, this risk was mitigated by the fact that relative measures were used rather than absolute measures (as...
described in section 3.2), as the average difference between two consecutive frames will then be much higher and the short-range correlations in the data considerably weaker as a result.

![LSTM Encoder-Decoder Future Predictor Model](image)

**Figure 4: LSTM Encoder-Decoder Future Predictor Model.** The input sequence (represented by the frames $v_1$ to $v_3$) is first fed into an encoder LSTM which creates a fixed size representation of that sequence using the weights contained in $W_1$. This learned representation is then used as the initial state for the decoder LSTM, which thereafter makes predictions for a fixed number of future timesteps (represented by the frames $v_4$ to $v_6$) using the weights contained in $W_2$. Each decoder LSTM cell is conditioned on the output from the previous decoder LSTM cell, i.e. the previous prediction $v_{t-1}$ is used to predict $v_t$.

An individual model was trained for each set of parsed data that was created with the methods described in section 3.2. Training parameters such as the learning rate and the number of epochs were shared between each compared model in the same way as the model parameters themselves. The input and output pairings used to train and validate each model were extracted from their respective set of parsed and normalized data. In which each index $n$ in a parsed and normalized sequence

$$F = \{f_1, f_2, \ldots, f_k\}$$

was converted into a unique input and output pairing $H_n$ as expressed by;

$$H_n = (\{f_i\text{ for } i = n - I - 1, \ldots, n - 1\}, \{f_o\text{ for } o = n, \ldots, n + O\})$$

where $I$ is the length of the input sequences and $O$ is the length of the output sequences. These pairings could then be randomly partitioned into either the training set or the validation set for the corresponding model. For convenience, models trained on event sequences and models trained on slice sequences are hereby denoted as event models and slice models, respectively.
3.3.1 Shared setup
The finalized shared model setup involved a single encoder and decoder layer with 200 hidden units each, an input length of 100 frames, and an output length of 10 frames. The data for each model was divided into a training and validation set following an 80/20 split, then each model was trained for 10 epochs on 500 000 input/output sequence pairs sampled with replacement from their respective training set. By fixing the training sample count, the same amount of computation time was allocated to the training of each model. The shared training sample count was chosen based on the maximum number of training samples available to any model, in order to avoid disrupting the distribution of the training sample sets. Moreover, training for more than 10 epochs did not manage to significantly improve the validation error for any of the trained models, with the validation error even increasing after a certain number of epochs for the models that used a temporal precision of 0.1 seconds or lower.

Limited tests with more than 200 hidden units were also performed. The increased number of hidden units primarily improving the validation error of the more temporally granular models as they were not as prone to overfitting. However, because of computational time limitations it was not possible to adequately train all the models when a larger number of hidden units were used, hence why these more complex models were not used for the experiments.

3.4 Evaluation
Evaluation of each parsing method was conducted in regards to primarily two properties taken from the corresponding prediction models, the mean-squared output error (MSE), and the mean temporal spacing (i.e. the amount of time between frames) of their input and output sequences. The mean-squared output error of a prediction model can be seen as a direct measure of the quality of its predictions. While the mean temporal spacing of its frames, both in its input and output, relates to how computationally expensive and memory intensive the model is to use, as a lower temporal spacing will mean that a larger number of frames will be needed to represent the same period of time and vice versa.

The following two sections will now detail the methods used for measuring these properties, as well as the method used for determining statistical significance of any observed differences.

3.4.1 Measurements
Measurements of the output error and the temporal spacing for each model were achieved by validating them on a number of shared validation sets. It was necessary to use the same validation set for each model as the resulting measures of output error and temporal spacing would otherwise not have been valid for comparison.

To avoid biasing the validation sets towards event-heavy time periods (i.e. simply test each model on their general prediction capabilities), all shared validation sets were created via the slice parsing method, with a slice spacing that was always greater than or equal to the
temporal granularity of all the models that would be tested on it. While the smallest possible time span for which a model can make frame predictions will be limited by the temporal granularity of its training data, there exists no hard limit for the largest prediction time span. Therefore, although it is impossible to, for example, validate a model trained on a temporal granularity of 0.1 seconds using a validation set with a temporal granularity of 0.01 seconds, the same is not true for the opposite scenario.

Every validation sample then consisted of a singular frame, hereby denoted as a validation slice, taken from one of these validation sets. Additionally, all samples were grouped together with a sequence of previously occurring frames in each of the different input formats used by the models. By feeding each of these sequences into the respective model they (the models) could individually be made to make predictions for the time period covered by a specific validation slice.

Thus, for each validation slice index \( v \) (with \( U \) as the validation slice spacing), and each model \( m \) (with \( I \) as the input length), an input sequence;

\[
P_{m,v} = \{ \vec{f}_i \text{ for } i = z_m - I, \ldots, z_m \} = \{ \vec{p}_{1}, \ldots, \vec{p}_{I} \}
\]

was extracted for that model and validation slice from the corresponding parsed sequence;

\[
F_{m} = \{ \vec{f}_{1}, \vec{f}_{2}, \ldots, \vec{f}_{k} \}
\]

where the index \( z_m \) of the frame closest in time to the validation slice was determined by;

\[
z_m = \text{argmax} \ K(z) = \{ \sum_{i=1}^{z} t_i | K(z) < U(v - 1) \}
\]

where \( t_i \) is the temporal spacing between the frames \( \vec{f}_i \) and \( \vec{f}_{i-1} \). Additionally, the difference \( T_v \) between the time of occurrence for the last frame \( \vec{f}_{z_m} \) and the time of occurrence for the respective validation slice is calculated and stored according to;

\[
T_v = U(v - 1) - \sum_{i=1}^{z} t_i.
\]

Now, because the frames contained measures of change rather than the absolute values for the variables, the temporal granularity and format of each models’ output could be changed to that used by the validation set by simply summarizing their denormalized prediction frames (i.e. output frames) into a prediction slice.

The expression for the creation of a prediction slice \( X_{m,v} \) for a specific model \( m \) (input length \( I \) and output length \( O \)) and a specific validation slice index \( v \) is therefore;

\[
X_{m,v} = \sum_{i=I+1}^{\infty} \vec{z}_i
\]

for which
\[ \bar{x}_i = \begin{cases} \bar{p}_i \otimes \bar{\sigma}_m + \bar{\mu}_m, & \text{if } U + T_v > \sum_{j=I+1}^{i} t_j \geq T_v \\ 0, & \text{otherwise} \end{cases} \]

where \( t_j \) is the temporal spacing between the frames \( \bar{p}_j \) and \( \bar{p}_{j-1} \), and each frame \( \bar{p}_i \) for \( i > I \) is created by the model \( m \) in batches of length \( O \) via the process:
\[
\{ \bar{p}_i, \ldots, \bar{p}_{i+O} \} = m(\{ \bar{p}_{i-I-1}, \ldots, \bar{p}_{i-1} \}) \{(i - I) \equiv 0 \ (\text{mod} \ O) \}
\]

After having calculated the prediction slice for each model and validation slice, all prediction slices were normalized via the process detailed in 3.2.2 using the mean \( \mu_Y \) and standard deviation \( \sigma_Y \) of the respective validation set \( Y \). It was then possible to make comparable measurements of the output error and temporal spacing associated with each parsing method. Firstly, the mean squared-error between a model’s prediction slice and the respective validation slice was used as the output error measure for that model and validation slice. While measures of the temporal spacing for a model in the context of a specific validation slice were in turn taken by averaging the temporal spacing of the frames contained in its input and output sequences.

Measurements of the mean-squared output error \( E_{m,v} \) for a specific model \( m \) and validation slice index \( v \) were thus taken by calculating:
\[
E_{m,v} = \frac{\sum_{r=1}^{R} E_{m,v,r}}{R}
\]
for which
\[
E_{m,v,r} = (X_{m,v,r} - Y_{v,r})^2
\]
where \( Y \) is the total validation set, \( Y_v \) is the validation slice at index \( v \) in the validation set \( Y \), \( X_{m,v,r} \) is the value of the variable \( r \) in the prediction slice \( X_{m,v} \), \( Y_{v,r} \) is the value of \( r \) in the validation slice \( Y_v \), and \( R \) is the number of variables (45 in this case).

The mean temporal spacing \( G_{m,v} \) for the input and output of each model \( m \) and validation slice index \( v \) was in turn calculated by the analogous formula:
\[
G_{m,v} = \frac{\bar{B}_{m,v}}{C_{m,v}}
\]
where
\[
\bar{B}_{m,v} = \sum_{i=1}^{i_E} b_i
\]
for which
\[ b_i = \begin{cases} 
   t_i, & \text{if } U + T_v > \sum_{j=i_S}^{i} t_j \geq T_v \\
   0, & \text{otherwise} 
\end{cases} \]

and

\[ C_{m,v} = \sum_{i=i_S}^{i_E} c_i \]

for which

\[ c_i = \begin{cases} 
   1, & \text{if } U + T_v > \sum_{j=i_S}^{i} t_j \geq T_i \\
   0, & \text{otherwise} 
\end{cases} \]

where \( i_S = 1 \) and \( i_E = I \) when calculating the mean temporal spacing for the input sequences (\( I \) being the input length for the model), and \( i_S = I + 1 \) and \( i_E = \infty \) when calculating the mean temporal spacing for the output sequences.

### 3.4.2 Statistical significance

The non-parametric Friedman test was used to determine the statistical significance of the differences between the different groups in regards to their mean-squared output error and mean temporal spacing, with the Wilcoxon signed-rank test being applied for post hoc analysis (with the Bonferroni correction to account for multiple comparison issues and using \( z/\sqrt{N} \) as the effect size measure). This was because the measurements of error and temporal spacing did not fulfill two of the necessary assumptions for repeated measures ANOVA, i.e. the assumptions of normality and homoscedasticity.
Chapter 4

Results

4.1 Training

Four different parsing methods were tested in total: parsing the data into regular event sequences (see section 3.2.1), parsing the data into rounded event sequences (section 3.2.1.1), and parsing the data into slice sequences (section 3.2.2) with two different values for the slice spacing, 0.01 seconds and 0.1 seconds (shortened to 0.01s and 0.1s respectively). The maximum value for the slice spacing was chosen based on the amount of data available, as a slice spacing of 1.0 seconds would have led to the creation of less than a 10,000 samples. In turn, the minimum value was chosen based on the average amount of time between the original events, which was roughly 0.01 seconds. The cut-off limits used were 1.0 seconds and 10.0 seconds, respectively, as these values represented the maximum amount of time that could be covered by a length-100 slice sequence using that spacing.

An Encoder-Decoder LSTM RNN (section 3.3) was then trained on the data resulting from each of these parsing methods. The results from training the final models are presented by the following plots (Figure 5) of their training and validation mean-squared-errors over time. For ease of comparison, the training and validation errors for each model have been normalized so that 1.0 represents the error obtained from simply using the mean as each prediction.

![Figure 5: Plots of the normalized training (red solid line) and validation errors (blue dashed line) as a function of the training epoch for each of the four models characterised by different data formats: (a) regular events, (b) rounded-events, (c) slices with spacing 0.01 seconds, and (d) slices with spacing 0.1 seconds.](image)
These results show that the 0.1 second slice model achieved the lowest average error on both its training and validation set, while the non-rounded event model, in turn, had the highest average errors. The other two models, i.e. the rounded event model and the 0.01s slice model, can be seen to have almost identical average errors on their training and validation sets, higher than the 0.1 second slice model but lower than the non-rounded event model.
4.2 Validation
As the goal for this thesis was to evaluate the effect that different types of parsing methods for event-based data have on the prediction accuracy and resources costs of RNNs applied to vehicular signal data, the following experiments were constructed in order to specifically measure those two properties in as wide a variety of situations as possible.

Two different validation sets, created from the same set of data files as was used for training, were used as the basis for these experiments, one with a slice spacing of 0.1 seconds (denoted as validation set 1) and the other with a slice spacing of 1.0 seconds (denoted as validation set 2). Each trained model was thereafter validated on all samples in each validation set following the process previously outlined in section 3.4. Note that for ease of comparison, each of the resulting measurements of error were normalized to a baseline of 1.0 (as mentioned earlier, the baseline corresponds to the model which simply always outputs the mean). Finally, the results for each group were compared via the use of the Friedman test and the Wilcoxon signed-rank test, with 0.01 as the desired p value.

The results of these tests are presented in two parts, the first section (4.2.1) concerning the prediction accuracy, and the second section (4.2.2) detailing the resource costs.
4.2.1 Prediction accuracy

Below are the results that concern the prediction accuracy of each model. To start, the Friedman test was conducted to compare the performance of multiple model configurations on two validations sets.

Table 1: Results from having applied the Friedman test to compare the mean-squared output errors for each model in the context of validation set 1 and 2. $\chi^2$ denotes the chi-squared test statistic, and $p$ denotes probability of these results given the null hypothesis (that the populations all have the same distribution).

<table>
<thead>
<tr>
<th>Friedman Test</th>
<th>Validation Set 1</th>
<th>Validation Set 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\chi^2$</td>
<td>190 510.05</td>
<td>9 865.18</td>
</tr>
<tr>
<td>$p$</td>
<td>&lt;0.001</td>
<td>&lt;0.001</td>
</tr>
</tbody>
</table>

The Friedman-test (Table 1) clearly shows that there are statistically significant differences between the different models concerning their output error for both validation sets, and it is therefore safe to move on with the post hoc analysis.

Table 2: Results from having used the Wilcoxon signed-rank test to make pairwise comparisons between the mean-squared output error of each model in the context of validation set 1 and 2. In the table, $z$ denotes the z-value, $p$ denotes the probability of these results given the null hypothesis (that the two compared populations have the same distribution), and $d$ denotes the effect size of the rank difference between the two groups (with a positive effect size indicating that group 1 generally had lower rankings than group 2 and vice versa).

<table>
<thead>
<tr>
<th>Wilcoxon Signed-Rank Test</th>
<th>Validation Set 1</th>
<th>Validation Set 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1</td>
<td>Group 2</td>
<td>z</td>
</tr>
<tr>
<td>Baseline</td>
<td>Regular Events</td>
<td>76.38</td>
</tr>
<tr>
<td>Baseline</td>
<td>Rounded Events</td>
<td>54.97</td>
</tr>
<tr>
<td>Baseline</td>
<td>Slices 0.01s</td>
<td>236.94</td>
</tr>
<tr>
<td>Baseline</td>
<td>Slices 0.1s</td>
<td>222.24</td>
</tr>
<tr>
<td>Regular Events</td>
<td>Rounded Events</td>
<td>111.76</td>
</tr>
<tr>
<td>Regular Events</td>
<td>Slices 0.01s</td>
<td>233.89</td>
</tr>
<tr>
<td>Regular Events</td>
<td>Slices 0.1s</td>
<td>229.76</td>
</tr>
<tr>
<td>Rounded Events</td>
<td>Slices 0.01s</td>
<td>226.64</td>
</tr>
<tr>
<td>Rounded Events</td>
<td>Slices 0.1s</td>
<td>229.19</td>
</tr>
<tr>
<td>Slices 0.01s</td>
<td>Slices 0.1s</td>
<td>18.27</td>
</tr>
</tbody>
</table>

Based on Table 2, we can assume that no two models had the same error distribution. The error distributions for each model are now shown below.
Figure 6: Boxplots (i.e. non-parametric distribution plots) of the normalized mean-squared-error for validation set 1 (left) and validation set 2 (right), displaying the results for the models in the following order: (a) baseline (green), (b) non-rounded event model (blue), (c) rounded event model (purple), (d) 0.01s slice model (orange), (e) 0.1s slice model (red). For each box, the box itself then shows the first and third quartiles for the temporal spacing, the whiskers the first and third quintiles, the thick line the median, and the small square the mean.

Figure 7: Bar graphs of the mean normalized mean-squared-error (1.0 representing the baseline) for each individual variable in regards to validation set 1 (left) and validation set 2 (right). In each group, the models’ bars are displayed in the following order: non-rounded event model (blue), rounded event model (purple), 0.01s slice model (orange), 0.1s slice model (red).
From Table 2 and Figures 6-7, it is possible to see that the 0.1s slice model managed to outperform the event models on all error measures, and that there was a similar result for the 0.01s slice model with a few notable exceptions. The primary exception is that the 0.01s slice model had a higher mean rank than the rounded event model on the second validation set (Table 2), with the caveat that neither model had a lower mean rank than the baseline. Additionally, there were also some specific variables for which the 0.01s slice model had a higher mean error than the event models (see variables 1 and 19 in Figure 7), albeit not to a statistically significant degree when considering the large amount of variables for which comparisons could be made and the low median error for each model on the individual variables.

Lastly, both the rounded event model and the 0.1s slice model achieved a significantly lower average error over the two validation sets than their more temporally precise counterparts, the non-rounded event model and the 0.01s slice model, respectively. Reducing the temporal precision of the data therefore managed to improve the prediction accuracy of both types of models. However, based on the fact that the rounded event model performed significantly worse than the 0.01s slice model on validation set 1, despite them both using the same temporal precision, it was not solely the temporal precision used for the data that influenced model performance.
4.2.2 Resource costs

Because the measurements of temporal spacing were practically identical for both validation sets, only the results for the first validation set (validation set 1) are shown here. Beginning with the results from the Friedman test.

Table 3: Results from having applied the Friedman test to compare the temporal spacing of each model’s input and output sequences in the context of validation set 1. $\chi^2$ denotes the chi-squared test statistic, and $p$ denotes probability of these results given the null hypothesis (that the populations all have the same distribution).

<table>
<thead>
<tr>
<th>Friedman Test</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\chi^2$</td>
<td>186 824.94</td>
<td>213 592.56</td>
</tr>
<tr>
<td>$p$</td>
<td>&lt;0.001</td>
<td>&lt;0.001</td>
</tr>
</tbody>
</table>

From Table 3 it can be concluded that there are statistically significant differences between the temporal spacing of both the input and output sequences for the two event models. Expanding upon this result with post hoc analysis gives Table 4.

Table 4: Results from having used the Wilcoxon signed-rank test to make pairwise comparisons between the mean temporal spacing of each event models’ input and output sequences in the context of validation set 1. In the table, $z$ denotes the $z$-value, $p$ denotes the probability of these results given the null hypothesis (that the two compared populations have the same distribution), and $d$ denotes the effect size of the rank difference between the two groups (with a positive effect size indicating that group 1 generally had lower rankings than group 2 and vice versa).

<table>
<thead>
<tr>
<th>Wilcoxon Signed-Rank Test</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group 1</td>
<td>Group 2</td>
<td>$z$</td>
</tr>
<tr>
<td>Regular Events</td>
<td>Rounded Events</td>
<td>245.26</td>
</tr>
<tr>
<td>Regular Events</td>
<td>Slices 0.01s</td>
<td>169.24</td>
</tr>
<tr>
<td>Regular Events</td>
<td>Slices 0.1s</td>
<td>204.80</td>
</tr>
<tr>
<td>Rounded Events</td>
<td>Slices 0.01s</td>
<td>227.86</td>
</tr>
<tr>
<td>Rounded Events</td>
<td>Slices 0.1s</td>
<td>201.38</td>
</tr>
<tr>
<td>Slices 0.01s</td>
<td>Slices 0.1s</td>
<td>290.60</td>
</tr>
</tbody>
</table>

Thus, the event models are shown to be pairwise different from one another in regards to the mean temporal spacing for both their input and output. The temporal spacing of the data belonging to each model is now visualized in Figure 8.
Figure 8: Boxplots of the mean temporal spacing for each model in regards to their training sequences (a-d), input sequences during validation (e-h), and output sequences during validation (i-l). In each category, the ordering is as follows; non-rounded event model (blue - a,e,i), rounded event model (purple - b,f,j), 0.01s slice model (orange - c,g,k), and 0.1s slice model (red - d,h,l). For each box, the box itself then shows the first and third quartiles for the temporal spacing, the whiskers the first and third quintiles, the thick line the median, and the small square the mean. Note that the boxplots of the slice models’ mean temporal spacing are degenerated since the mean temporal spacing for a slice sequence will always be equal to the slice spacing (see section 3.2.2).

It is possible to notice in Figure 8 that the non-rounded event model and the 0.01s slice model had the lowest temporal sparsity for their input and output (where a low temporal sparsity is defined as the data having a low mean for its temporal spacing and vice versa). The non-rounded event model and the 0.01s slice model were therefore required to make more predictions and have a longer input sequence (if the input sequence is to represent the same amount of time) than the respective rounded event model and 0.1 second slice model, and as such were more resource intensive to use. Both using a lower precision for the timestamps in the event sequences and increasing the slice spacing in the slice sequences thus resulted in a reduction in resource costs.

Additionally, as also shown in Figure 8, the rounded event sequences had a higher temporal sparsity than the slice sequences that used a slice spacing of 0.01 seconds, despite both having a temporal precision of 0.01 seconds. Therefore, when using the same level of temporal precision for both types of parsing methods, the event parsing method led to a considerably more compact representation of the data.

Finally, it should be noted that because the validation targets, unlike the training targets, were evenly sampled over the temporal domain, the mean temporal spacing for the two event models’ input sequences varied greatly between training and validation. However, the temporal spacing of the event models’ output more closely followed the distribution of the training input.
Chapter 5

Discussion

5.1 Summary of key findings
It has been shown that parsing vehicular data as an input to an LSTM based prediction model in the form of evenly spaced signal slices instead of event-based representations considerably improved the models’ overall prediction accuracy. However, provided that the temporal precision was kept the same, having an event-based rather than slice-based data format resulted in a more compact representation of the original dataset and thus lower resource costs.

Additionally, it was demonstrated that the temporal precision used for the data was the major determining factor behind the resource costs of using the resulting model, with a higher temporal precision leading to increased costs and vice versa. Reducing the temporal precision of the original data representations was, up to a point, also shown to decrease the model’s output error. Therefore, the choice of which temporal precision to use for the data was a critical factor in determining the resulting performance of the prediction model.

5.2 Analysis of results
The primary failing of the event models is most likely that they were unable to capture salient patterns of temporal correlations relevant to the problem domain, as was partially evidenced by the difference in temporal distribution between the event-based models’ input and output sequences. This hypothesis is further supported by how an event and slice model were demonstrated to have different validation errors despite having almost identical training errors, since that means that the failure of the event models coincided with the temporal variable receiving increased importance. Considering that the choice of representation for the temporal element was the primary difference between the two model types, this failing is also the most likely reason for the generally poor performance of the event models (when compared to the slice models).

However, while the slice models had an overall higher prediction accuracy than the event models there were indications that there are some advantages to using an event model over a slice model provided that they are both trained to work at a similar level of temporal precision. The rounded event model, for example, managed to slightly outperform the corresponding slice model on some variables, and event representations were in general shown to be more compact than slice representations. The second advantage is, however, as would be expected since slice sequences are simply event sequences for which the temporal variables have been replaced by the addition of zero-set frames.
5.3 Relation to other works

Due to the lack of pre-existing research on the subject of parsing event-based data into an RNN model, it is difficult/impossible to consider these findings in the context of relevant previous works. However, as a stand-alone study, this thesis can serve as a basis for future work on this topic. Clearly, it should be in the interest of Scania AB to convert their event-based vehicular signal data into slice-based data if their goal is to use it to train RNNs, and the same is assumed to be true for other, similar, application domains where the goal is to train an RNN using event-based data. It would also seem to be beneficial for Scania AB to reduce the temporal precision of their data before using it to train an RNN. Though this specific conclusion is not extrapolatable to other application domains, as it is extremely dependent on the exact setup of the data in question. We also note the possibility that there exists a more complex event-based representation that is effective to use for RNNs, as our results only strongly suggest that the most straightforward event-based representation is not an effective choice. Furthermore, based on these results as well as the general predominance of slice-based representations in the context of applying machine learning to event-based data, it would seem likely that the advantages of such a representation extend to other types of machine learning models than just RNNs. Otherwise, one would expect more literary examples of unevenly spaced representations being used for event-based data, especially considering the ad-hoc way in which the representation is most often chosen.

Overall though, RNNs definitely seem to be applicable to the analysis of the multivariate vehicular signal data, as our models were capable of significantly outperforming the baseline despite being severely limited in size. There is thus promise in using RNNs and vehicular signal data for other tasks than just improving testing efficiency through the extraction of domain knowledge. The most obvious one is monitoring/diagnostics, wherein the vehicular signal data could, for example, be used for the early detection of engine problems. However, both the network and the dataset will need to be specifically tailored for each such end application. This is because there is still no one-size-fits-all approach to neural networks, despite their reduced reliance on hand-engineered features. As an example, even though many different model setups were tested over the course of this project, we have still not come close to determining the optimal hyperparameters. In addition, it can also be difficult to devise an output-target that is appropriate for accomplishing a specific task. Hence, the frequent development of new types of targets within the field of neural networks. In fact, the next step for Scania AB will be to address this type of issue, as the domain knowledge contained in a trained prediction model is not directly accessible.

Moreover, as with other sets of real-world multivariate time series data, choosing a reasonable set of input variables from the vehicular signal data proved to be very important. Although the methods we used were crude and mostly manual, they still significantly improved both the performance and accuracy of our models. However, while we managed to reduce the number of prospective variables to a reasonable amount (i.e. a number that is possible to sift through by hand) by simply sorting out those that had way too few
occurrences in the data files, this would not have been the case if the dataset had been significantly bigger. Thus, it would likely be beneficial to replace these simple methods for variable selection with more advanced such methods in the future, with ICA as one of the potential candidates. Using more advanced methods for variable selection would also have the additional benefit of reducing redundancy and complexity in the dataset.

5.4 Limitations and possible improvements
The amount and scope of the experiments performed as part of this project were primarily limited by two factors: dataset size and time. As a result, it was not possible to establish an upper bound for the model complexity, i.e. the optimal number of hidden units and layers, for each model that would have resulted in the highest accuracy, nor to have the input sequences for each model cover the same amount of time, hence biasing the results towards the less resource intensive models. Additionally, these limitations also made it necessary to use the same set of data for validation as was used for training, thus introducing an additional bias towards the less resource intensive models, as they will generally be more prone to overfitting (precautions were however taken to prevent the models from overfitting). However, many of the observed differences were convincing given the considerable effect size, and it thus seems rather unlikely that they are solely the result of the aforementioned bias.

Reflecting on the present work, there is some room for improvement, specifically in relation to the core methodology. The primary improvement would be the inclusion of a separate test set for the model comparison step. The addition of a second set of experiments for which the input sequences for each model are made to cover the same amount of time is a second potential improvement. Both of these actions would serve to increase the confidence of the results, as well as to enable a more in-depth analysis. Implementation choices made early on in the project are the reason why these changes were not included in this specific study, as fixing these issues would have required large-scale changes to the code as well as a complete retraining of all models at the point where they had been recognized. On a minor note, it would also have been beneficial to train a second rounded event model, one that uses a temporal precision of 0.1 seconds rather than 0.01 seconds, since the set of rounded event models would then have mirrored the set of slice models (with respect to their temporal precisions).

5.5 Ethics and sustainability
Improving the efficiency and rigor of the vehicle tests at Scania AB will indirectly increase the safety of their vehicles and potentially reduce their emissions as well. As it will then be possible for Scania AB to perform more tests on the fuel efficiency and safety of their vehicles in the same amount of time. However, the event-based vehicular signal data risks containing sensitive identifying information, such as GPS-data, and it is important that such information is cleansed from the data before it is to be used for analytical purpose.
Chapter 6

Conclusions and future work

In conclusion, parsing the vehicular signal data into a prediction model as slice-based data vastly improved the prediction accuracy of the model when compared to parsing it as event-based data (its original format), although with the minor drawback of slightly increasing the resource costs for using the model. Furthermore, reducing the temporal precision of the data as a part of the parsing process was proven to markedly affect both the prediction accuracy and the associated resource costs of the resulting prediction model, with a moderate reduction of the precision improving both measures in our case.

Although the results obtained from this study are informative, there is definitely room for a more extensive study to be done on the subject in the future, especially if that study involves another application domain than vehicular signal data. Future work could also focus on developing entirely new methods for parsing / handling event-based data in its original format, which then might avoid the problems identified in this study while retaining the advantages of using that data format.
Bibliography


