Context by Proxy: Identifying Contextual Anomalies Using an Output Proxy

Jan-Philipp Schulze
jan-philipp.schulze@aisec.fraunhofer.de
BMW Group & Fraunhofer AISEC
Munich, Germany

Artur Mrowca
artur.mrowca@bmw.de
BMW Group
Munich, Germany

Elizabeth Ren
ren@isi.ee.ethz.ch
ETH Zürich
Zurich, Switzerland

Hans-Andrea Loeliger
loeliger@isi.ee.ethz.ch
ETH Zürich
Zurich, Switzerland

Konstantin Böttinger
konstantin.boettinger@aisec.fraunhofer.de
Fraunhofer AISEC
Garching, Germany

ABSTRACT

Contextual anomalies arise only under special internal or external stimuli in a system, often making it infeasible to detect them by a rule-based approach. Labelling the underlying problem sources is hard because complex, time-dependent relationships between the inputs arise. We propose a novel unsupervised approach that combines tools from deep learning and signal processing, working in a purely data-driven way. Many systems show a desirable target behaviour which can be used as a proxy quantity removing the need to manually label data. The methodology was evaluated on real-life test car traces in the form of multivariate state message sequences. We successfully identified contextual anomalies during the cars' timeout process along with possible explanations. Novel input encodings allow us to summarise the entire system context including the timing such that more information is available during the decision process.

CCS CONCEPTS

• **Computing methodologies** → **Anomaly detection**; *Unsupervised learning*; • **Information systems** → *Data mining*.

KEYWORDS

anomaly detection; unsupervised learning; recurrent neural networks; signal processing; automotive

ACM Reference Format:
Jan-Philipp Schulze, Artur Mrowca, Elizabeth Ren, Hans-Andrea Loeliger, and Konstantin Böttinger. 2019. Context by Proxy: Identifying Contextual Anomalies Using an Output Proxy. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3292500.3330780

KDD '19, August 4–8, 2019, Anchorage, AK, USA

ACM ISBN 978-1-4503-6201-6/19/08…$15.00
https://doi.org/10.1145/3292500.3330780

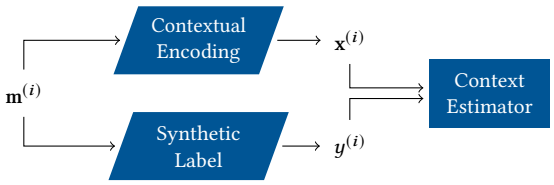
1 INTRODUCTION AND RELATED WORK

Modern vehicle systems become increasingly more complex, involving the communication and operation of a multitude of software and hardware components. Common cars comprise 100 million lines of source code on-board, and up to 15 controllers communicating per function with 2 million messages exchanged per minute. To verify such systems, massive diagnostic traces modelled as multivariate event sequences are recorded to be analysed off-board. For example, BMW Group's fleet of 500 test cars produced around 1.5 TB per day in 2017. Due to the massive amount of recorded data, these traces become increasingly difficult to analyse using existing methods. Contextual dependencies and correlations among the input dimensions impede the detection of unknown misbehaviour. Therefore automatic anomaly detection methods for multivariate temporal event sequences become important.

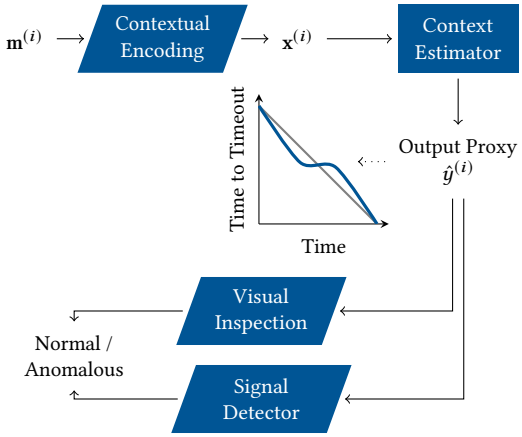
Known errors (e.g. erroneous states) can be detected by handmade rules and statistics. However, detecting unknown faults in multivariate correlating event data requires to take the context of the overall system at the time of the fault occurrence into account. These so called *contextual anomalies* manifest only under certain conditions depending on intrinsic (e.g. the state of the engine) or extrinsic (e.g. the weather) factors, as well as on the timing between state changes in the system. In a complex system, such as a car, it is usually infeasible to manually detect contextual anomalies. However, it is often possible to judge whether a subsystem behaves as expected by looking at a particular proxy quantity. In this work, we use the example of a car's timeout process, i.e. its transition to a state of least energy consumption. Whenever the customer turns off the engine and leaves the car, only vital system parts should remain active to save battery charge. Within a correct context, i.e. when a valid sequence of events with the right timings occurs, the proxy *time to the next timeout* has a desired behaviour: the timeout should be approached in every time-step.

By exploiting these insights, we developed a deep learning approach for contextual anomaly detection. As depicted in Fig. 1, the basic idea is to train a recurrent neural network (RNN), called the *context estimator*, based on historic system data. We encoded the overall systems states – intrinsic, extrinsic and timing – as inputs by a suitable *context encoder*. As output, we introduced synthetic labels by a proxy quantity, called the *output proxy*. With this, the context estimator is able to estimate the relationship between the

system context and the output proxy. With new data as input, we used the context estimator to predict the expected proxy curve. By comparing the predicted to the expected curve (e.g. a linearly decreasing one), deviations indicate anomalies. Dominant factors for the anomaly can be determined by analysing the signals that are active at the points of deviation. With this, our method allows to detect contextual anomalies and pinpoint the context that caused the anomaly.



(a) Training phase: the impact of the historic system context on the output proxy is learned.



(b) Detection phase: the output proxy based on the current system context is estimated and checked for anomalies.

Figure 1: The contextual anomaly detector extracts the system context $\mathbf{x}^{(i)}$ from the bus message $\mathbf{m}^{(i)}$ at each unique time instance $t^{(i)}$. Anomalies are detected when the assumptions about the estimated output proxy $\hat{y}^{(i)}$ are broken.

1.1 Related Work

Anomaly detection is of great importance in several use cases. Data points that show different behaviour than others might lead to error cases or other points of interest. Research on anomaly detection is distinguishable by three main points: the data type, the available labels and the general class of the underlying learning algorithm. The surveys by Chandola et al. [3, 4] and Agrawal et al. [2] provide a good overview about the wide range of choices. Our work is based on multivariate state message sequences where no anomaly-related labels are available. However, by introducing an output proxy, we are able to use an RNN architecture combined with signal processing methods to find anomalous examples.

A common example of anomaly detection on sequential message-based systems is the analysis of network traffic, e.g. IP packets. Here, anomalous behaviour could lead to points where the infrastructure has been attacked. There are different ideas how to handle this data type. The sequential data might be summarised in hand-crafted features. For example, Lu et al. [14] use average values of connection statistics such that wavelet analysis can be applied to it. Another approach might be to generate an objective function based on the data. Shon et al. [29] weight the importance of the components of IP packets such that an SVM can be used to separate anomalous examples. In each of these approaches, the content of the sequential messages must be manually weighted. Our approach is purely data-driven: the neural network is given an unfiltered input vector. Thanks to a special input encoding made for sequential data, the algorithm learns to select suitable inputs among over 1000 dimensions.

Multiple data mining methods for diagnosis of faults and anomalies in automotive systems were introduced in the past. These include feature-based fault detection methods, such as clustering approaches [6, 8, 13, 16] or neural networks [12, 18]. Other methods used condition indicators [27] or rule-based approaches [1] for this task. Further, fault classification was applied in various automotive scenarios, such as for the diagnosis of the adaptive cruise control [20]. Such approaches use neural networks [10, 19], decision trees [20], SVMs [25] and automated analysis frameworks for inspection [17]. Of equal importance is the application of data mining to solve prediction tasks, e.g. to predict engine [32] or compressor faults [26], allow for predictive maintenance [21] or estimate the remaining useful life of vehicular batteries [28, 30].

In this paper, we use RNNs with long short-term memory (LSTM) elements as first proposed by Hochreiter & Schmidhuber [9]. LSTMs are well suited for sequential data and have been applied to anomaly detection in message-based systems. Malhotra et al. [15] use LSTM elements to predict the next entries in a time series based on the training of a normal series. Similar approaches have been leveraged to other applications. O'Shea et al. [22] have applied anomaly detection using LSTMs on radio communication, Chauhan et al. [5] on electrocardiography. In all aforementioned approaches the LSTM is trained on valid samples. The error between the prediction and the sample is compared; in case it is anomalously high, an anomaly is detected. Our research follows a different approach due to two main reasons. Firstly, in our case it is infeasible to guarantee that the training data is free from anomalies as pre-production traces are used. Secondly, using an error function would shift the anomaly detection to finding a suitable threshold on the error. We thus propose a model that learns the relationship between the input messages and their impact on a synthetic output proxy. Our approach bundles the contextual information of the entire message series in an easy-to-grasp quantity distinguishing between normal and anomalous behaviour.

1.2 Contributions

In the scope of this work, we make the following contributions:

- Design, training and validation of a data-driven anomaly detector for contextual anomalies
- Design of input encodings for sequential message-based data
- Adaptation of a recursive line-fit algorithm for sparse message-based data

2 PRELIMINARIES

2.1 Timeout Process

We inspect real-life data of pre-release cars provided by a big automobile company. Multiple information dimensions, called *signals*, are transmitted over time on communication busses. Each signal determines the state of a specific system part, e.g. the current speed, the engine state or the driving mode. The process we are analysing for anomalies is called the *timeout process*, which is the system's transition to a state of least energy consumption. There is no fixed sequence of signals, but several combinations of state transitions result in a valid timeout process. Intrinsic (e.g. voltage levels), or extrinsic (e.g. user inputs) events may prolong this process. Some prolongations are legitimate (e.g. a user input) while others indicate anomalies. Under normal circumstances, the system should approach the state of least energy consumption with each time-step once the timeout process started. An appropriate method for contextual anomaly detection needs to mark instances where the overall system context leads to a timeout process that breaks this assumption.

Vehicles are in three main states: active, paused and inactive. When *active*, most system components are turned on. As the customer turns off the engine, the system becomes *paused*. In this state, comfort features like the air conditioning may still be used. Once the customer has left the vehicle, most system components are turned off until the *inactive* state is reached. Here, the point at which the timeout occurs is registered. In this state, the vehicle consumes the least energy as only vital system components (e.g. the anti-theft system) are active. It is important that the inactive state is reached eventually as otherwise battery charge is wasted, which may result in customer dissatisfaction and high warranty costs for the company.

2.2 Input Data Type

Recording the bus communication yields a multivariate state sequence described by N message vectors $\mathbf{m}^{(i)}$ for time-step $i \in \{1, \dots, N\}$. Each message is uniquely characterised by its three components: *time*, *origin* and *value*. The time component relates to the time the message was sent. It is not unique as messages from different origins may be sent at the same time instance. Each system part is identified by the origin of the signal, which defines the type of information transmitted. Based on this type, the message carries values of different data types. These are *categorial* values, which describe a discrete state that the system is currently in, and *numerical* values containing continuous information. Combinations of both data types are also possible.

The messages are compressed within the company for long-term availability. Compression of the raw messages involves the following steps: repetitions are removed, numerical values with high value ranges are clustered and quantities derived from multiple signals are added. In detail, messages with the same origin-value pair are not repeated over time, numerical signals that fluctuate around discrete numerical values are grouped in categorial values and e.g. warnings based on value thresholds are added to the data. Our approach is applied on the compressed data. An example of such a data set is given in Table 1a.

2.3 Linear State Space Filters

Parts of our proposed contextual anomaly detector rely on model-based signal processing methods using linear state space models (LSSMs). LSSMs give a mathematical description of systems where an output is generated based on inputs and an inner state. Wildhaber et al. [31] use LSSMs to build recursive signal detectors where previous calculations are reused. We chose this efficient algorithm as a step towards on-board diagnostics where computing resources are usually limited. In the following we present the main aspects of this approach with the example of line-fitting. Assuming equally spaced samples, a line can be described by the state vector $\mathbf{a} = (a_0, a_1)^\intercal \in \mathbb{R}^2$. These two parameters represent the y-intercept and the slope of the line when plotted in Euclidean space. For time-step i, the predicted output of the system will be $\hat{y}_i = a_0 + a_1 \cdot i$. This system can be represented in state space form using the state-transition matrix $\mathrm{A} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$ and the output vector $\mathbf{c} = \begin{pmatrix} 1 & 0 \end{pmatrix}$:

$$\mathbf{a}_i = \mathrm{A} \cdot \mathbf{a}_{i-1}$$
$$\hat{y}_i = \mathbf{c} \cdot \mathbf{a}_i$$

Fitting this model to available data points y_i corresponds to the process of finding the model parameters $\hat{\mathbf{a}}$ that minimise the error between the prediction and the observed samples. We will focus on an open left-sided model, i.e. where only samples left of the current one are taken into account. In practice, data points far away from the current time instance should influence the fit less such that the model adapts to the changed environment. For this reason, a general window function $\alpha_{i,k} \in \mathbb{R}$ is introduced located around index k. A popular choice is the exponentially decaying window with decay rate η, i.e. $\alpha_{i,k} = \eta^{k-i}$. Some data points may be ignored completely, hence a weight $w_i \in [0, 1]$ is introduced. Overall, the objective of the fitting process becomes:

$$\hat{\mathbf{a}}_k = \underset{\mathbf{a}}{\operatorname{argmin}} \sum_{i=1}^{k} \alpha_{i,k} w_i (y_i - \hat{y}_i)^2 = \underset{\mathbf{a}}{\operatorname{argmin}}\, J_k$$

This objective is fulfilled when minimising the error J_k with respect to the parameters $\mathbf{a}$, i.e. requiring $\nabla_{\mathbf{a}} J_k = 0$. Using the parametrisation $J_k = \mathbf{a}^\intercal \mathrm{W}_k \mathbf{a} - 2\mathbf{a}^\intercal \xi_k + \kappa_k$, the estimated parameters are given by $\hat{\mathbf{a}}_k = \mathrm{W}_k^{-1} \cdot \xi_k$. Recursive formulae can be found for effective calculations as given in the original paper [31].

3 METHODOLOGY

Our proposed contextual anomaly detector consists of two main parts: a context estimator and a signal anomaly detector as shown in Fig. 1b. Both parts are joined by a synthetic quantity which we refer to as the *output proxy* as introduced in Sec. 3.2. It summarises the high-dimensional input to a quantity of known behaviour. The context estimator is an RNN that learns to estimate the impact of the current system context on the output proxy. For this, we developed an input encoding that summarises all information given on the communication bus as presented in Sec. 3.1. Given new data, the context estimator is used to estimate the most likely output proxy value. The trajectory of the output proxy is then automatically assessed in the signal anomaly detector to reveal where the current system context violates the expected behaviour. For our use case, we found the *time to the next timeout* to be a suitable output

proxy as a main characteristic is known: the timeout should be approached in every time-step during the timeout process. The detection of anomalies is automated by the slope detector as presented in Sec. 3.4.

3.1 Input Encoding

Contextual anomalies only arise under certain system contexts, i.e. if certain intrinsic or extrinsic events occur at a specific timing. For cars, the system context is implicitly given by the communication bus messages, e.g. the minimal example shown in Table 1a. All components of the messages – time, origin, value – may lead to an anomalous situation. Thus, a contextual anomaly detector must capture the impact of each of these components. In the following, we present four encoding schemes that summarise the entire system context as input to the context estimator.

Basic Scheme. All introduced encodings have the same number of features. Each origin-value pair corresponds to one feature in the encoding, i.e. one dimension in the input vector. If the origin is of categorial type, each possible value of this category is represented as a separate feature. Also, each numerical value corresponds to a feature for each origin. This feature scheme allows the context estimator to weight the respective importance without human bias. In our case, the input comprises 1530 origin-value pairs. At each time-step i of the recording, a new feature set is generated, while entries that occur at the same time instance $t^{(j)} = t^{(i)}, i \neq j$ are accumulated into one feature vector. This combines all changes which may influence the system context at each unique time instance.

1-Hot. A common way to represent categorical data is the 1-Hot encoding. In a N-categorical classification system, category x is encoded by the value 1 at position x in a N-dimensional vector. This encoding can be extended to multi-categorical systems by allowing more than one 1 in the entire vector. In our context, this means that at each time instance $t^{(i)}$, all active signal origins are set to 1 at the respective position. 1-Hot produces a highly sparse input as shown in Table 1b.

1-Hold. To enhance 1-Hot for representing sequential data, we introduce the 1-Hold encoding. Instead of dropping the information about previous messages, the last known value for each signal origin is retained. For categorical entries, the feature component of the new origin-value pair is set to 1, while all the other values of this signal origin are reset to 0. In contrast, numerical values are retained as long as no new value is sent. 1-Hold leads to less sparse input vectors as it contains all system states up to the current time-step.

γ-Replace. Timing is crucial in vehicle systems such that it should be encoded in the input. For example, the late arrival of a signal value for a state change might cause anomalous behaviour in system components that are awaiting this particular information. We introduce a novel encoding scheme called γ-Replace that unlike 1-Hot and 1-Hold includes the temporal history of previous messages. States and values of previous events are kept, but their influence fades with longer temporal distance. Intuitively, past information becomes less likely to represent an important part of the system behaviour at a current time instance, which we solve as follows.

Table 1: Example of the possible input encodings.

(a) Example message series.

i	Time	Origin	Value
1	10 s	Light Status	Front On
2	20 s	Speed	10-30 km/h
3	20 s	Steering Angle	40 Degree
4	25 s	Light Status	All Off
5	28 s	Light Status	Front On

(b) 1-Hot encoded example sequence.

Light Status		Speed	Steering Angle
Front On	All Off	10-30 km/h	Numerical
1	0	0	0
0	0	1	40
0	1	0	0
1	0	0	0

(c) γ-Additive encoded example sequence.

Light Status		Speed	Steering Angle
Front On	All Off	10-30 km/h	Numerical
1	0	0	0
$1 \cdot \gamma^{\frac{10}{60}}$	0	1	40
$1 \cdot \gamma^{\frac{15}{60}}$	1	$1 \cdot \gamma^{\frac{5}{60}}$	40
$1 \cdot \gamma^{\frac{18}{60}} + 1$	$1 \cdot \gamma^{\frac{3}{60}}$	$1 \cdot \gamma^{\frac{8}{60}}$	40

Again, categorical values that are set in the current message are encoded by 1 at the respective origin-value position, while numerical entries are retained. However, instead of resetting all other categorical values, an exponential decay factor γ is applied based on the current temporal gap. With this method, the time distance between two messages is implicitly encoded by the amount of decay. We will define γ to be the decay per minute, i.e. a value $f(t_0) = c$ at time t_0 decays in 60 s to $f(t_0 + 60) = \gamma \cdot c$. Denoting time difference as $\Delta t^{(i+1)} = t^{(i+1)} - t^{(i)}$, the feature vector $\mathbf{x}^{(i)}$ only containing categorical data evolves according to:

$$\mathbf{x}^{(i+1)} = \gamma^{\frac{\Delta t^{(i+1)}}{60}} \cdot \mathbf{x}^{(i)}$$

γ-Additive. A remaining disadvantage of γ-Replace is the loss of information about how often categorical components were set. This means that by replacing the current origin-value pair with 1, it cannot be known whether this component was set before. By adding 1 to feature components that are reoccurring, a feature > 1 shows that this category has been set before. We will thus introduce γ-Additive as extension to γ-Replace where the respective components are added, instead of being replaced by 1. An example is given in Table 1c.

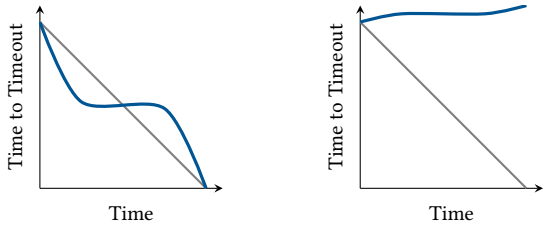
3.2 Output Proxy

Problems arise when describing contextual anomalies as they may be too subtle to detect or too complex to be grasped in rules. Hence,

labelling the data of the timeout process quickly becomes infeasible when done manually. We solve this issue by considering an implicit target quantity for which a known desirable behaviour can be deduced at any time. This quantity will be called the *output proxy*. The output proxy is used as a synthetic label which the context estimator is trained on along with the current system context. If the estimated output proxy differs from the target behaviour, an anomaly is found.

In the case of timeout processes, we found the *time to the next timeout* to be a suitable output proxy. There is no fixed timeout sequence, but many possible signal combinations exist that prolong or accelerate the timeout process. However, a target behaviour is known: the timeout is expected to be approached on every consequent recorded message. In timeout processes, we identified two patterns of the output proxy that define an anomaly. These are long plateaus and rising predictions as shown in Fig. 2. The former indicates delayed timeouts, while the latter indicates prolongations. By using the output proxy, both the location and the cause of the anomaly are found when analysing the underlying bus messages.

A successful timeout occurs if no message is recorded for a defined amount of time, i.e. the communication buses are silent. The timeout value t_{timeout} is given as a flag in the data. Starting from the first message in the state of interest, i.e. the paused state, the labels are calculated as $y^{(i)} = t_{\text{timeout}} - t^{(i)}$ (in seconds) for each recorded message along with the encoded input. With this, both a feature vector $\mathbf{x}^{(i)}$ and a corresponding label $y^{(i)}$ are determined. As illustrated in Fig. 2, the training data resembles a straight line stating the time to the next timeout at every point in time.



(a) Plateau **(b) Rising**

Figure 2: Possible training data (grey) and anomalous behaviour (blue) during the prediction of the output proxy.

Plateaus and Rises. Plateaus result from incoming messages that do not alter the timeout process. Hence, a system part prolongs the timeout process. A rise in the timeout prediction is related to messages that have a negative impact on the next timeout. Both behaviours may be valid for short time ranges if the timeout is approached again afterwards. For example, if a user presses a button to activate the air conditioning, the system enables the respective system components in order to provide the functionality to the user; the timeout will likely prolong as all these system parts must be deactivated before the car becomes inactive. We design the anomaly detector to accept short corrections in the time to the next timeout, but no significant rises for longer time periods. Thanks to the output proxy, we have shifted the problem of finding contextual anomalies to interpreting an easy-to-grasp quantity, i.e. the time to the next timeout. This process can easily be automated by a suitable signal anomaly detector.

3.3 Neural Network Architecture

Due to their high model capacity, neural networks (NN) are powerful models for learning correlations from labelled data. With the output proxy as label and the encoded data as input, we place special requirements on the NN. Firstly, it should not imitate, i.e. overfit, the training data, but learn a general relationship between the input messages and the output proxy. This requires an architecture that leaves enough freedom for the parameter sets to evolve over time. In order to preserve temporal relationships between the input and the output, we use an RNN. In particular, such that also long-term temporal dependencies are captured, we make use of LSTM units. After qualitatively evaluating different combinations of architectures, we propose a three-layered NN that consists of two RNNs and one feed-forward NN leading to a scalar output. The output dimensions of the RNN layers, 256 and 384 respectively, were determined in a non-exhaustive grid search. This architecture gave smoother results than shallower networks. Further, Pascanu et al. showed in [23] that multi-layered RNNs have a positive influence on the prediction capabilities.

Loss Function. We used the mean squared error (MSE) for computing both the training and the validation loss. In particular, our goal is to introduce flexibility in the model for generalisation, i.e. the model should be able to diverge from the training labels to a certain extent. However, the prediction of normal samples should still follow a general trend: the intended behaviour in the shape of a decreasing time to the next timeout. The MSE is a good choice to achieve both goals: small deviations between the training and predicted data lead to relatively low errors, whereas large deviations are amplified by the quadratic nature of the MSE.
Note that the MSE is the wrong function to measure the quality of the model: as it is desirable to have a model that diverges from the labels, lowering the loss does not necessarily correspond to a better prediction. We strongly aim for generalisation rather than overfitting. Only in this case, the model has learned to weight the impact of the system context on the output proxy. Imitating the training labels themselves, i.e. a straight line, would imply, each system origin influenced the timeout process by the same extent. Consequently, high values in the MSE are expected and desirable.

3.4 Slope Detection

To reveal contextual anomalies in the timeout process, we inspect the time to the next timeout estimated based on the current system context. We identified two patterns in this output proxy that lead to anomalous behaviour: plateaus and rising predictions. This process can be done manually by visual inspection, but is easily automated with the proposed LSSM-based slope detector. The two undesirable patterns correspond to small (positive) gradients over an extended range of time. Slight adaptations to the presented slope detection algorithm must be made as uniformly sampled signals are expected. For easier interpretability, we calculate the decay η using the time resolution t_{res} and the decay per minute γ. The time resolution determines the minimal time two messages must be separated such that they are distinguishable in time, effectively acting as the new

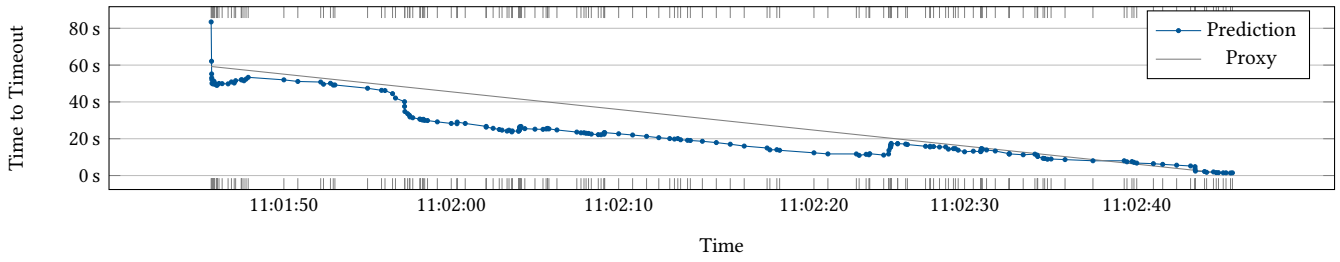


Figure 3: Normal timeout process where the general, downward-facing target behaviour of the output proxy is visible.

sampling period. The number of samples per minute is calculated as $n_{\text{samples}} = \frac{60}{t_{\text{res}}}$. Hence, the decay factor becomes $\eta = \gamma^{\frac{1}{n_{\text{samples}}}}$. To include the absolute time scale, virtual zero-weighted samples are added between the messages that have no effect on the line-fit. The time instances of the actual samples are adjusted to the nearest sample point on the new uniform time scale. For zero-weighted samples, the recursion becomes $\mathrm{W}_{k+1} = \eta \mathrm{A}^{-\mathsf{T}} \mathrm{W}_k \mathrm{A}^{-1}$. Assuming that the next sample appears after N time-steps, the virtual samples are taken into account within one recursion step by noting:

$$\begin{aligned} \mathrm{W}_{k+N-1} &= \eta \mathrm{A}^{-\mathsf{T}} \mathrm{W}_{k+N-2} \mathrm{A}^{-1} \\ &= \eta \mathrm{A}^{-\mathsf{T}} \eta \mathrm{A}^{-\mathsf{T}} \mathrm{W}_{k+N-3} \mathrm{A}^{-1} \mathrm{A}^{-1} \\ &= \ldots = \eta^{N-1} (\mathrm{A}^{-\mathsf{T}})^{N-1} \mathrm{W}_k (\mathrm{A}^{-1})^{N-1} \end{aligned}$$

For two messages separated by the time difference Δt and hence the number of virtual samples $N_{\text{virt}} = \left\lfloor \frac{\Delta t}{t_{\text{res}}} \right\rfloor$, the recursion becomes:

$$\begin{aligned} \mathrm{W}_{k+1} &= \eta^{N_{\text{virt}}} (\mathrm{A}^{-\mathsf{T}})^{N_{\text{virt}}} \mathrm{W}_k (\mathrm{A}^{-1})^{N_{\text{virt}}} + w_{k+1} \cdot \mathbf{c}^{\mathsf{T}} \mathbf{c} \\ \xi_{k+1} &= \eta^{N_{\text{virt}}} (\mathrm{A}^{-\mathsf{T}})^{N_{\text{virt}}} \xi_k + w_{k+1} \cdot \mathbf{c}^{\mathsf{T}} \cdot y_{k+1} \\ \kappa_{k+1} &= \eta^{N_{\text{virt}}} \kappa_k + w_{k+1} \cdot y_{k+1}^2 \\ \nu_{k+1} &= \eta^{N_{\text{virt}}} \nu_k + w_{k+1} \end{aligned}$$

This recursion can efficiently be applied to the output proxy. Furthermore, we added a mechanism to ignore the first (usually volatile) samples. This is simply done by setting $w_k = 0$ for the respective time-steps. We ignore the first 0.5 s and chose a low $\gamma = 10^{-10}$ to focus on recent samples. The time resolution is at 10 ms for smoother results. When manually observing the output proxy, we look for instances where our assumption of a decreasing time to the next timeout is violated. Translated to the parameter estimate $\hat{\mathbf{a}}_k$, an anomaly is found when the slope is above $-10^{-2.5}$ for 30% of the entire sample time, i.e. when the time to the next timeout does not change or rises for a long time.

3.5 Overall Method

By combining the aforementioned theories, the overall method is summarised by the following steps as depicted in Fig. 1:

(1) **Training Phase.** During training, the context estimator, i.e. the RNN estimating the system context, is conditioned on the available training data $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{\text{train}}$. At each unique time instance $t^{(i)}$ the corresponding system context is encoded in the input vector $\mathbf{x}^{(i)}$. In particular, all relevant extrinsic, intrinsic and temporal influences on the output proxy are incorporated. The corresponding label $y^{(i)}$ is calculated based on the output proxy, i.e. the time to the next timeout.

(2) **Detection Phase.** At each unique time instance, the context estimator predicts how the current system context influences the output proxy, i.e. $\hat{y}^{(i)}$ is estimated. This results in a curve, which can be inspected visually. To automate this process, the signal anomaly detector is used to reveal the time instances where anomalous deviations between the estimated and the expected behaviour are present. In order to reveal the cause of anomalies, the underlying messages that led to the timeout violation can be analysed.

With this method, automated detection of contextual anomalies in timeout processes of vehicles was performed. Experts are efficiently pinpointed to potential points of interest. With this, costs in both time and effort can be reduced.

4 EVALUATION

The proposed methodology was evaluated on real-life test vehicle data. In the following, we provide examples that motivate the validity of our method. As encoding, we used a 0.1-Additive scheme as motivated in Sec. 4.2. 4343 randomly selected timeout processes were used for the training of the context estimator.

4.1 Normal Example

Before looking at anomalous examples, it is instructive to familiarise oneself with a normal sample as shown in Fig. 3. Each dot corresponds to a new message on the bus system and hence a new prediction of the output proxy. It can well be seen that the general trend has successfully been learned: the timeout is gradually approaching. Three intriguing points are visible: the beginning, a downward correction past 11:01:50 and a rise past 11:02:20. The volatile start is well explained by the architecture: information only flows forward in time. Thus, little about the context is known at the first message. However, the input encoding allowed an estimate not far from the later predictions.

More intriguing is the first downward correction. Here, the context estimator proposes that the timeout comes closer based on the past messages. Analysing the underlying bus messages shows that the driver's door was closed at this instance. This is a remarkable result as the model has predicted a behaviour that is intuitive to human observers: if the driver's door was closed, the driver has likely left the vehicle such that it can enter the inactive state. The messages at the increasing prediction indicate that at least one door

is unlocked and the light is on. These system parts have to be turned off before a timeout can happen.

This example demonstrates that the model was able to learn intuitive relationships between the vehicle's context and the time to the next timeout. Results like these show that – although no ground truth is available – the predictions follow a relatable scheme. The general trend to an approaching timeout was clearly shown. Deviations are well justified by observing the underlying messages. These are powerful features when analysing traces: unusual behaviour can quickly be observed and explained. We like to stress that these results were achieved by a purely data-driven model, no human bias has been added.

4.2 Input Encoding

Having developed different encodings for sequential message-based data, it is instructive to compare their impact on the prediction quality. In Fig. 4 an overview of the validation losses is provided. A striking feature is the extremely high MSE. In usual regression problems, i.e. where the training labels should be matched exactly, this would be an alarming sign. However, since we aim for a model that learns to weight the influence of each message, it is required that the prediction leaves the path of the synthetic training labels. Thus, the plot is mainly a motivation that the model was able to improve from the random initialisation. It can be seen that the two decay-based encodings are the best scoring models in the non-overfitting regime. In contrast, 1-Hot causes the model to be volatile right from the beginning. For better comparison, we thus show the predictions of a common sample in Fig. 5.

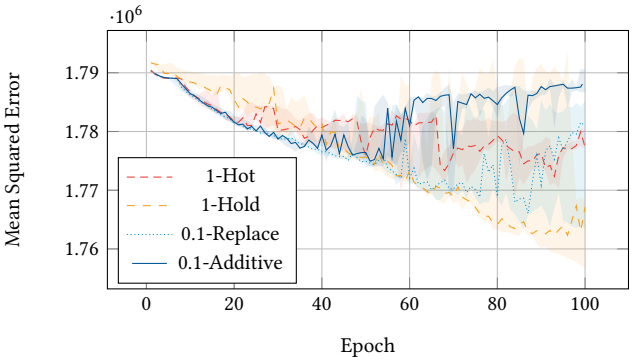


Figure 4: Median and upper/lower validation loss for different input encodings based on three repetitions on 329 validation samples. Overfitting starts roughly after the 50$^{\text{th}}$ epoch.

1-Hot. 1-Hot produces the most volatile prediction, which is also the furthest away from the proxy. At multiple instances, the prediction jumps to the the timeout level of the synthetic data. This behaviour seems to arise from the highly sparse input vector 1-Hot produces. Corrections are abrupt as no fine-graded context information is given. Some of the corrections seem to imitate behaviour learned in the training data. The time for the model to grasp the context takes longer: in the first seconds, 1-Hot causes predictions that are further away from the test data than for 0.1-Additive. These characteristics underline the volatile validation loss.

0.1-Additive. According to the validation loss and the predictions, the decay-based encodings produce a more stable outcome close to the proxy. We argue that a reliable model is of great importance in the search of anomalies in data; a too volatile prediction could induce false positives in the detection process. As the additive encoding contains the most information, this encoding is used for the final model. Obviously, using the MSE as loss function cannot grab the qualitative differences in the prediction – an issue open for future work.

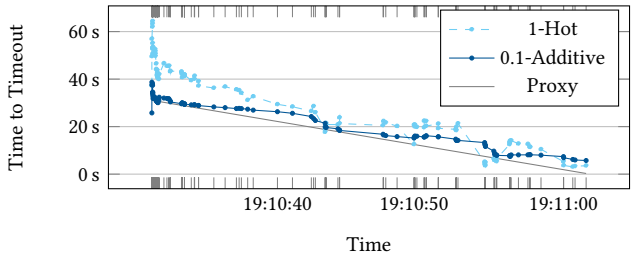


Figure 5: Predictions for models using the same training samples, but different input encodings, 1-Hot and 0.1-Additive, for a normal test sample.

4.3 Anomalies

4.3.1 Synthetic Anomalies. Contextual anomalies do not follow a general scheme such that it is hard to manually detect them. Consequently, only a few real-life examples exist. In a first step, we want to motivate the validity of our method by introducing artificial anomalies in the test data. We do so by inserting random signals, i.e. perturbations that are likely out-of-context. For this, we take samples with a minimal length of 30 s where previously no anomaly was detected and add a categorial signal. The results as a function of the amount of inserted time-steps and inserted signals per time-step are presented in Table 2 where 100 signals have been added at random time instances during the first 30 s of the trace.

Table 2: Number of detected anomalies among 100 synthetically added random bus messages as a function of the number of inserted time-steps and signals.

Added Time-steps \ Added Signals	1	2	3	4	5	6
1	26	24	40	43	50	53
2	28	52	62	64	62	67
3	38	65	63	76	70	82
4	49	67	65	79	82	86
5	59	77	80	79	81	78
6	64	76	81	83	84	92

It can be seen that already small out-of-context insertions result in positive anomaly detection using our method. The more perturbations are added, the more likely it is that an anomaly is detected. This behaviour is well expected, as it also becomes more likely that a signal is out-of-context the more insertions are made. Of course,

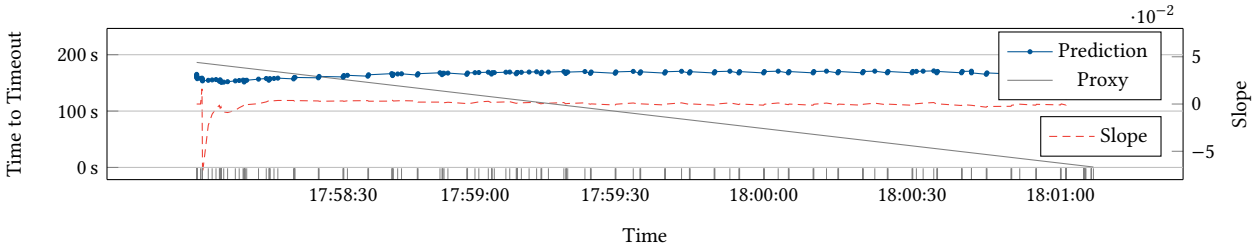


Figure 6: Context estimation of a known contextual anomaly showing a plateau in the output proxy.

the severity of the inserted signal is dependent on which signal is added at which time instance. The detection results can be adapted by the signal detector parameters, i.e. for how long and by what extent the output proxy may violate the target behaviour.

This synthetic example shows that the context estimator has successfully learned to weight the importance of each signal in the overall context. Random signal insertions are a good example of likely out-of-context behaviour. Our method is already able to detect small perturbations and shows more confidence the more insertions are made. In the following, real-life examples are discussed which show a less random behaviour and thus make detection using our method easier.

4.3.2 Known Anomaly. A manually detected contextual anomaly has been added to the test data, i.e. a sample the context estimator has not seen before. The respective plots for the prediction of the output proxy and its slope are shown in Fig. 6. According to the error description, messages repeated for several minutes cause the timeout process to prolong. These messages originated from the same system part which was not allowed to interrupt the timeout by that time range. Even by visual inspection, this sample looks anomalous in comparison to all other shown examples. Besides the first seconds of the prediction that are ignored during the detection, the slope is always above zero; in the beginning, the prediction increased until a plateau is formed. The model seems to have learned that the messages prolong the timeout process and thus are anomalous when repeated for a long time. This example underlines the generalisation capabilities of the model; even though it was not trained to find this anomaly in particular, its prediction does not imitate the training data but shows out-of-normal behaviour.

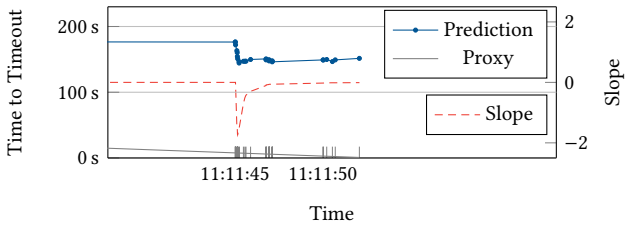


Figure 7: Excerpt of a discovered contextual anomaly.

4.3.3 Discovered Anomaly. During the research, recent data was evaluated as shown in Fig. 7. One sample revealed an intriguingly long plateau causing the timeout to postpone. In consultation with the respective engineer, this anomaly was recently discovered. Ethernet messages illegitimately wake up the vehicle at periodic time instances. The model successfully predicted that these messages prolong the time to the next timeout.

5 DISCUSSION

5.1 Lessons Learned

A major limitation of most of today's deep learning approaches is the need for a massive amount of labelled data. Often, considerable time and work is invested in generating qualitative labels that suit the desired application. It quickly became obvious that labelling contextual anomalies with their highly interrelated functional dependencies and timings is infeasible for us. Our work has overcome this issue by introducing a synthetic label: the output proxy. Although the proxy does not provide information whether a sample is anomalous, in our case it contains a measurement that leads to instances of anomalous behaviour. In that, a system was trained on synthetic data, but can be used to decide about real-life inputs. We hope to spark the interest of research in how synthetic labels can be used in the training process of future intelligent systems.

Additionally, we have introduced multiple input encodings for sequential message-based data. They were designed with the goal of summarising the input dimensions at hand, i.e. time, origin and value. We have empirically shown that a qualitative improvement can be seen when using these encodings. Input encodings for NNs are a topic that has barely been touched in research. We could qualitatively show that intelligently summarising the input information can improve the overall system capabilities.

5.2 Future Work

In the scope of this work, promising results were achieved on contextual anomaly detection. A main challenge was to build a model without knowing a ground truth. By using a synthetic output proxy, quantifying the error of predictions requires different measurements than in usual regression/classification problems. Future work may look at which loss functions best describe the prediction error and how this impacts the model training. Ideas that come to mind are loss functions weighting the error differently based on the time, or to introduce a similarity measure.

The proposed method was designed for timeout processes, but should be flexible enough for other detection cases. Currently, the time to the next timeout is used as the output proxy. Other applications might reveal other suitable proxy quantities. Future research may leverage these results to a more general setting.

6 CONCLUSION

In this work, a contextual anomaly detector was designed and evaluated on real-life test car data. Thanks to the output proxy, the context estimator summarised a high-dimensional input system in one easy-to-grasp quantity. Anomalies may be found by visual inspection, which can be automated by a suitable signal anomaly detector. The underlying causes of anomalies are revealed by observing the messages that caused fluctuations in the output proxy. Thanks to our method, experts are efficiently pinpointed to potential points of interest. With this, costs in both time and effort can be reduced.

ACKNOWLEDGMENTS

This paper greatly benefited from cross-functional and comprehensive collaboration between the departments "system verification electrics/electronics" and "data analytics connected vehicle" at the *BMW Group Research and Development Centre.*
Special thanks go to *Robert Schuster* who intensely supported us by pushing this use case, sharing his technical experience with us and by enabling us to work and research on excellent preprocessed valuable data sets.

REFERENCES

[1] Puneet Agarwal, Gautam Shroff, Sarmimala Saikia, and Zaigham Khan. 2015. Efficiently discovering frequent motifs in large-scale sensor data. In *Proceedings of the Second ACM IKDD Conference on Data Sciences - CoDS '15*. ACM Press, New York, New York, USA, 98–103. https://doi.org/10.1145/2732587.2732601

[2] Shikha Agrawal and Jitendra Agrawal. 2015. Survey on Anomaly Detection using Data Mining Techniques. *Procedia Computer Science* 60, 1 (2015), 708–713. https://doi.org/10.1016/j.procs.2015.08.220

[3] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2009. Anomaly Detection: A Survey. *Comput. Surveys* 41, 3 (2009), 1–58. https://doi.org/10.1145/1541880.1541882

[4] Varun Chandola, Arindam Banerjee, and Vipin Kumar. 2012. Anomaly Detection for Discrete Sequences: A Survey. *IEEE Transactions On Knowledge and Data Engineering* 24, 5 (2012), 823–839. https://doi.org/10.1109/TKDE.2010.235

[5] Sucheta Chauhan and Lovekesh Vig. 2015. Anomaly detection in ECG time signals via deep long short-term memory networks. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*. 1–7. https://doi.org/10.1109/DSAA.2015.7344872

[6] M.Y. Chen, E. Kiciman, E. Fratkin, A. Fox, and E. Brewer. 2002. Pinpoint: problem determination in large, dynamic Internet services. In *Proceedings International Conference on Dependable Systems and Networks*. IEEE Comput. Soc, 595–604. https://doi.org/10.1109/DSN.2002.1029005

[7] François Chollet and Others. 2015. Keras. https://keras.io.

[8] J.A. Crossman, Hong Guo, Y.L. Murphey, and J. Cardillo. 2003. Automotive signal fault diagnostics - part I: signal fault analysis, signal segmentation, feature extraction and quasi-optimal feature selection. *IEEE Transactions on Vehicular Technology* 52, 4 (jul 2003), 1063–1075. https://doi.org/10.1109/TVT.2002.807635

[9] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation* 9, 8 (1997), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735 arXiv:1206.2944

[10] Hong Guo, J.A. Crossman, Y.L. Murphey, and M. Coleman. 2000. Automotive signal diagnostics using wavelets and machine learning. *IEEE Transactions on Vehicular Technology* 49, 5 (2000), 1650–1662. https://doi.org/10.1109/25.892549

[11] Diederik P. Kingma and Jimmy Lei Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6 (2014). arXiv:1412.6980v9 http://arxiv.org/abs/1412.6980

[12] Kyusung Kim and A.G. Parlos. 2002. Induction motor fault diagnosis based on neuropredictors and wavelet signal processing. *IEEE/ASME Transactions on Mechatronics* 7, 2 (jun 2002), 201–219. https://doi.org/10.1109/TMECH.2002.1011258

[13] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. 2003. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery - DMKD '03*. ACM Press, New York, New York, USA, 2–11. https://doi.org/10.1145/882082.882086

[14] Wei Lu and Ali A Ghorbani. 2008. Network Anomaly Detection Based on Wavelet Analysis. *EURASIP Journal on Advances in Signal Processing* 2009, 1 (jun 2008), 837601. https://doi.org/10.1155/2009/837601

[15] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. 2015. Long Short Term Memory Networks for Anomaly Detection in Time Series. In *ESANN*. 89–94.

[16] Artur Mrowca, Barbara Moser, and Stephan Günnemann. 2019. Discovering Groups of Signals in In-Vehicle Network Traces for Redundancy Detection and Functional Grouping. In *Machine Learning and Knowledge Discovery in Databases*. Springer International Publishing, 86–102. https://doi.org/10.1007/978-3-030-10997-4_6

[17] Artur Mrowca, Thomas Pramsohler, Sebastian Steinhorst, and Uwe Baumgarten. 2018. Automated Interpretation and Reduction of In-Vehicle Network Traces at a Large Scale. In *2018 55th ACM/ESDA/IEEE Design Automation Conference (DAC)*. IEEE, 1–6. https://doi.org/10.1109/DAC.2018.8465937

[18] Tobias Carsten Müller. 2010. Komplexitätsreduzierte neuronale Netze zur Offboard-Diagnostik in Fahrzeugsystemen / Tobias Carsten Müller. (2010). https://publikationsserver.tu-braunschweig.de/receive/dbbs_mods_00033996

[19] Yi L. Murphey, M. Abul Masrur, and ZhiHang Chen. 2006. Fault Diagnostics in Electric Drives Using Machine Learning. Springer, Berlin, Heidelberg, 1169–1178. https://doi.org/10.1007/11779568_124

[20] Oliver Niggemann, Benno Stein, Thomas Spanuth, and Heinrich Balzer. 2009. Using Models for Dynamic System Diagnosis: A Case Study in Automotive Engineering. In *5. Dagstuhl-Workshop MBEES 2009: Modellbasierte Entwicklung eingebetteter Systeme*. 46–56. https://www.hs-owl.de/skim/opus/frontdoor/index/index/docId/3135

[21] Slawomir Nowaczyk, Rune Prytz, and Stefan Byttner. 2012. Ideas for Fault Detection Using Relation Discovery. In *The 27th annual workshop of the Swedish Artificial Intelligence Society (SAIS); 14-15 May 2012; Örebro; Sweden*. 49. http://www.ep.liu.se/ecp_home/index.en.aspx?issue=071

[22] Timothy J O'Shea, T. Charles Clancy, and Robert W. McGwier. 2016. Recurrent Neural Radio Anomaly Detection. *CoRR* abs/1611.0 (2016). arXiv:1611.00301 http://arxiv.org/abs/1611.00301

[23] Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to Construct Deep Recurrent Neural Networks. *CoRR* abs/1312.6 (2013), 1–13. arXiv:1312.6026

[24] F Pedregosa, G Varoquaux, A Gramfort, V Michel, B Thirion, O Grisel, M Blondel, P Prettenhofer, R Weiss, V Dubourg, J Vanderplas, A Passos, D Cournapeau, M Brucher, M Perrot, and E Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[25] T. Praveenkumar, M. Saimurugan, P. Krishnakumar, and K.I. Ramachandran. 2014. Fault Diagnosis of Automobile Gearbox Based on Machine Learning Techniques. *Procedia Engineering* 97 (2014), 2092–2098. https://doi.org/10.1016/J.PROENG.2014.12.452

[26] Rune Prytz, Sławomir Nowaczyk, Thorsteinn Rögnvaldsson, and Stefan Byttner. 2015. Predicting the need for vehicle compressor repairs using maintenance records and logged vehicle data. *Engineering Applications of Artificial Intelligence* 41 (may 2015), 139–150. https://doi.org/10.1016/J.ENGAPPAI.2015.02.009

[27] Ioannis A Raptis, Christopher Sconyers, Rodney Martin, Robert Mah, Nikunj Oza, Dimitris Mavris, George J Vachtsevanos, and Ioannis Raptis. 2013. A Particle Filtering-based Framework for Real-time Fault Diagnosis of Autonomous Vehicles. In *Annual Conference of the Prognostics and Health Management Society 2013*.

[28] Bernhard Schlegel, Artur Mrowca, Peter Wolf, Bernhard Sick, and Sebastian Steinhorst. 2018. Generalizing application agnostic remaining useful life estimation using data-driven open source algorithms. In *2018 IEEE 3rd International Conference on Big Data Analysis (ICBDA)*. IEEE, 102–111. https://doi.org/10.1109/ICBDA.2018.8367659

[29] Taeshik Shon, Yongdae Kim, Cheolwon Lee, and Jongsub Moon. 2005. A machine learning framework for network anomaly detection using SVM and GA. *Proceedings from the 6th Annual IEEE System, Man and Cybernetics Information Assurance Workshop, SMC 2005* 2005 (2005), 176–183. https://doi.org/10.1109/IAW.2005.1495950

[30] Sergii Voronov, Daniel Jung, and Erik Frisk. 2016. Heavy-duty truck battery failure prognostics using random survival forests. *IFAC-PapersOnLine* 49, 11 (jan 2016), 562–569. https://doi.org/10.1016/J.IFACOL.2016.08.082

[31] Reto A. Wildhaber, Nour Zalmai, Marcel Jacomet, and Hans-Andrea Loeliger. 2018. Windowed State-Space Filters for Signal Detection and Separation. *IEEE Transactions on Signal Processing* 66, 14 (2018), 3768–3783. https://doi.org/10.1109/TSP.2018.2833804

[32] Peter Wolf, Artur Mrowca, Tam Thanh Nguyen, Bernard Bäker, and Stephan Günnemann. 2018. Pre-ignition Detection Using Deep Neural Networks: A Step Towards Data-driven Automotive Diagnostics. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 176–183. https://doi.org/10.1109/ITSC.2018.8569908

A IMPLEMENTATION DETAILS

A.1 Network Architecture

During our research, we compared different architectures qualitatively. Architectures involving feed-forward NN were only able to capture the desired output pattern (e.g. a decreasing time to the next timeout) when given less training samples. Simple RNN networks, involving one LSTM element, already had enough capacity to give successful results. Nonetheless, the proposed architecture involving two LSTM elements gave smoother predictions with less fluctuations. More LSTM elements did not result in severe changes, such that the shallower architecture was preferred. Their respective hidden state dimensionality was selected based on a non-exhaustive grid search summarised in Table 3. The best validation loss after 20 epochs was used for the architecture decision.

Table 3: Grid search results for the RNN architecture (L1-regulariser $\lambda = 0.01$, dropout at 10%).

		LSTM2		
		128	256	384
LSTM1	128	1,230,349	1,227,336	1,226,097
	256	1,230,050	1,227,193	1,225,266
	384	1,230,342	1,227,746	1,225,820

Adam as introduced by Kingma et al. in [11] was used as optimiser with its default values. The RNN model had a small L1-regularisation factor of $\lambda = 0.01$ on the LSTM weight matrices. Dropout was at 10% in between the recurrent layers. For the implementation of the NN, we used the Keras [7] framework. We would like to give the output of Keras `model.summary()` function that summarises the layers used in Keras. Note that the dropout layers are implicitly given in the LSTM architecture by setting the corresponding parameters.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
masking_3 (Masking)          (None, 209, 1530)         0
_________________________________________________________________
lstm_5 (LSTM)                (None, 209, 256)          1829888
_________________________________________________________________
lstm_6 (LSTM)                (None, 209, 384)          984576
_________________________________________________________________
time_distributed_3 (TimeDist (None, 209, 1)            385
=================================================================
Total params: 2,814,849
Trainable params: 2,814,849
Non-trainable params: 0
_________________________________________________________________
```

A.2 Input

For generating the input, we selected test car traces at random. We filtered the data past October 2017 for one car model. A trace comprises messages in the order of 10^6; several timeout processes are among these. Whenever the messages showed that the car is in the paused state, the input was recorded along with the calculated output proxy until the timeout is registered (announced by a special message). We will call the encoded data, i.e. the set of feature vectors, for each timeout process a *sample*. The length of the samples is not constant, but usually around 100 feature vectors. Note that there might be cases where the timeout is not announced during the paused state. The samples of these traces are ignored as this is anomalous behaviour.

Inputs for RNNs in Keras are expected to be a 3-dimensional tensor in the format $(N_{\text{samples}}, N_{\text{time-steps}}, N_{\text{features}})$. N_{features} was known in advance by crawling through parts of the data in the paused state and noting all seen origin-value pairs. N_{samples} is based on how many traces were processed and how many timeout processes were available. However, $N_{\text{time-steps}}$ is not fixed in our data. We decided on manually fixing $N_{\text{time-steps}}$ to the 95^{th} percentile of all available time-step lengths: 209 in our case. This was done by clipping samples with more time-steps accordingly, while all other samples were padded. Such that only real data, not the padded values, are used for training, Keras allows to introduce so called *Masking Layers*. We padded with a masking value of -1; if the entire feature vector consists of this value, the time-step is not used for training. Unusually long samples could also be used as a static anomaly detector, i.e. where an anomaly is detected solely based on a threshold. However, we have seen long valid traces and thus recommend using the proposed contextual anomaly detector.

A.3 Preprocessing

During our research, we used several combinations of the preprocessors available in SciKit Learn [24]. As most of the proposed input encodings range in $[0, 1]$, we tried applying the `MinMaxScaler` on the numerical data only. Here, all values are scaled in $[0, 1]$ based on their minimum and maximum value. Unfortunately, this gave inferior results where the prediction was merely a constant. Surprisingly, applying the `StandardScaler` to all features regardless of their original range, gave the best results. Applying the `StandardScaler` to the numerical values only and the `MinMaxScaler` to the categorial, did not improve the results. Usually, `StandardScaler` should be applied to normally distributed data – which is not given for our numerical data.

A.4 Model

The models used in this work were trained on timeout processes of randomly sampled traces available in the data set. For the input encoding comparison, the models were trained on the very same samples using the respective encoding. If not stated otherwise, a model trained on 4343 samples encoded by 0.1-Additive was used. For the input comparison, 2956 samples using the respective encoding were used. The same samples were used for all encodings to allow comparisons.