Embedded Hybrid Anomaly Detection for Automotive CAN Communication

Marc Weber, Simon Klug, Eric Sax, Bastian Zimmer

Embedded Hybrid Anomaly Detection for Automotive CAN Communication

Marc Weber, Simon Klug
and Eric Sax
Institute for Information Processing Technologies
Karlsruhe Institute of Technology
Karlsruhe, Germany
Email: marc.weber3@kit.edu, simon.klug@student.kit.edu
and eric.sax@kit.edu

Bastian Zimmer
Vector Informatik GmbH
Stuttgart, Germany
Email: bastian.zimmer@vector.com

***Abstract*—Due to the steadily increasing connectivity combined with the trend towards autonomous driving, cyber security is essential for future vehicles. The implementation of an intrusion detection system (IDS) can be one building block in a security architecture. Since the electric and electronic (E/E) subsystem of a vehicle is fairly static, the usage of anomaly detection mechanisms within an IDS is promising. This paper introduces a hybrid anomaly detection system for embedded electronic control units (ECU), which combines the advantages of an efficient specification-based system with the advanced detection measures provided by machine learning. The system is presented for - but not limited to - the detection of anomalies in automotive Controller Area Network (CAN) communication. The second part of this paper focuses on the machine learning aspect of the proposed system. The usage of Lightweight On-line Detector of Anomalies is investigated in more detail. After introducing its working principle, the application of this algorithm on the time series of a CAN communication signal is presented. Finally, first evaluation results of a prototypical implementation are discussed.**

***Keywords*—intrusion detection system, anomaly detection, machine learning, LODA, automotive, Controller Area Network, time series**

I. Introduction

Today, connectivity and highly automated driving are the two major topics pushing the evolution of automotive electronics. Both enable a significant improvement for passenger comfort and safety. However, especially in their combination, connectivity and highly automated driving yield new dangerous scenarios. On the one hand, vehicles become increasingly connected with their environment and other vehicles. New wireless technologies like WiFi, Bluetooth and Car2X communication are installed, which enable new cyber-attack vectors [1][2][3]. On the other hand, ECUs get more and more control over safety-relevant functions of a vehicle, like braking and steering, in order to realize automated driving. To counter the risk of fatal cyber-attacks, several researchers and leading companies propose a multi-layer security concept [1][4][5][6][7]. A so-called defense in depth architecture could e.g. consist of four defense barriers as proposed by Miller and Valasek [1]:

1) Secure off-board communication
2) Access control for in-vehicle networks
3) Separation of different domains within the electric and electronic architecture
4) Mechanisms to detect and prevent cyber-attacks on vehicle networks and within ECUs

The paper at hand focuses on the last defense barrier, for which related research and industry propose the installation of IDS and intrusion prevention systems (IPS) [1][7][8][9]. This is especially true for in-vehicle CAN networks, since CAN is the most important and most frequently used automotive bus system at the moment. The presented IDS concept for CAN combines the efficiency of a specification-based approach with the advanced detection of irregularities using machine learning algorithms. Although the IDS system is introduced for automotive ECUs, the concept is not limited to that use case. The system is designed in a flexible way, enabling an easy adaption to different network technologies, as well as ECUs in different application areas.

Primary goal of the presented system is improving the cyber security of ECUs. As one part of the last defense barrier, it checks for irregularities in vehicle networks and it does not rely on pre-defined attack patterns. With this approach, it is possible to recognize also unknown and newly arising cyber-attacks. However, the detection mechanisms work independent of the root cause of an irregularity, which potentially is a cyber-attack but also could be a malfunction of an ECU. For the automotive domain, the latter is especially relevant for the next generation of vehicles, if we consider the upcoming self-adapting and machine learning-based vehicle functions. The proposed system could help safeguarding these functions during runtime since a complete validation at development time becomes difficult.

The remaining part of this paper is structured as follows: Section II introduces the term *anomaly* and its different types, followed by the discussion of related work in section III. The first central aspect is the elaboration of the proposed hybrid anomaly detection system in section IV, starting with a general system overview and continuing with an explanation of the single building blocks. The second part of

this paper starts with section V, which discusses the topic of anomaly detection in time series data with special emphasis on automotive communication signals. With respect to the proposed system and its boundary conditions, Lightweight On-line Detector of Anomalies is selected to be evaluated in more detail for this use case. Therefore, section VI introduces its working principle, followed by the description of its application on CAN communication signals in section VI-A. This second central aspect of the paper at hand is further elaborated in section VII by presenting first evaluation results. Section VIII concludes this paper with a summary and gives an outlook on future work regarding the hybrid anomaly detection system.

II. Different Types of Anomalies

Irregularities, so-called anomalies, are deviations from normal behavior. In literature, an anomaly is also referred to as outlier, besides others. D. M. Hawkins gives a corresponding definition, which reflects the basic assumption of the proposed system [10]:

"An outlier is an observation that deviates so much from other observations as to arouse suspicion that it was generated by a different mechanism."

The following paragraphs introduce different types of anomalies, which are useful to classify anomaly detection mechanisms. The simplest form of an anomaly is the point anomaly, defined by Chandola et al. as follows [11]:

"If an individual data instance can be considered as anomalous with respect to the rest of data, then the instance is termed as a point anomaly."

Exemplarily for the automotive context, the available data could be a CAN bus log. In this case, the mentioned data instance is a single recorded CAN message. If that single message can be classified as anomaly without considering other data instances, e.g. because it contains the information that the vehicle is driving 200 km/h at 1500 rpm in the 1st gear, it is a point anomaly. In contrast to the point anomaly, a contextual anomaly can only be classified as such, if additional contextual information is taken into account [11]. Having a look at the example above, if a CAN message contains the data 250 km/h at 4000 rpm in the 6th gear, this would not be a point anomaly since it is realistic in the first place. However, if the message was recorded while driving inner city, it can be declared as a contextual anomaly. The last type is the collective anomaly, referring to a sequence of data instances, which together form an anomaly [11]. This would e.g. be the case if two or more consecutive CAN messages indicate an unrealistic or physical impossible acceleration of the vehicle.

III. Related Work

Besides defining the different anomaly types, Chandola et al. did a comprehensive state of the art analysis for anomaly detection techniques [11]. Many of them are based on machine learning algorithms, having the advantage that the behavior of the observed system has not to be known in advance but the normal behavior is learned from training data. Machine learning is powerful but also has some disadvantages. Although most algorithms are successfully applied in different application domains, including intrusion detection, for some of them it is hard to understand what happens in detail inside the system, e.g. when using large neural networks. In consequence, it is hard to prove that machine learning algorithms work as intended under all circumstances. Additionally, most of them are resource intensive in terms of required computing power and/or storage consumption. Therefore, their application within embedded software is currently limited and has to be evaluated with care.

As mentioned previously, researchers and industry experts recommend the usage of an IDS for automotive CAN communication [1][7][8][9]. However, there is one crucial difference compared to the classical application areas of IDS solutions in computers and computer networks: In-vehicle networks are used in a well-defined environment. E.g. all exchanged CAN messages, sent and received by ECUs, are defined by the vehicle manufacturers in advance to guarantee interoperability. This means that there is a lot of knowledge available, defining the normal behavior of the bus system. Additionally, the automotive industry has established several standards to specify the communication between ECUs in a semi-formal manner, often referred to as communication matrix. This specification includes the exchanged messages, which are distinguished based on their identifiers, and the transported payload. The payload itself consists of signals, each representing a single transported data element, e.g. vehicle speed, rpm and gear. Figure 1 illustrates a CAN frame with an exemplary payload. Re-using the available knowledge in an anomaly detection system for CAN communication is vital. This is also reflected in the discussion of anomaly detection for in-vehicle networks by Müter et al. [12]. After discussing signature- versus anomaly-based detection mechanisms for IDS in the automotive context, they argue in favor of anomaly-based detection. For signature-based systems, the need for regular updates and the focus on known attack patterns are presented as the main disadvantages. Although the first disadvantage gets smaller or will even disappear in future due to the possibility of remote updates, the focus on known attacks remains and is inherent to the principle of signature-based detection. Therefore, to further discuss anomaly-based detection is still valid today. Müter et al. mention the disadvantage of a high false positive rate for most anomaly detection mechanisms. This is not acceptable for a system installed in a vehicle. As a consequence, they present an anomaly detection concept which does not produce false positives at all. Their proposed system is based on the assumption that the normal behavior of vehicle networks can be defined successfully, which supports the statement that re-using OEM knowledge is important. In the remainder of their paper, Müter et al. introduce eight classes of network monitors - so-called anomaly detection *sensors*. Figure 2 shows the different sensor classes defined in [12]. Furthermore, the authors discuss the applicability of the different sensors and summarized their results in Figure 3. The realization of these sensors in context of the proposed hybrid anomaly detection system is further elaborated in section IV.

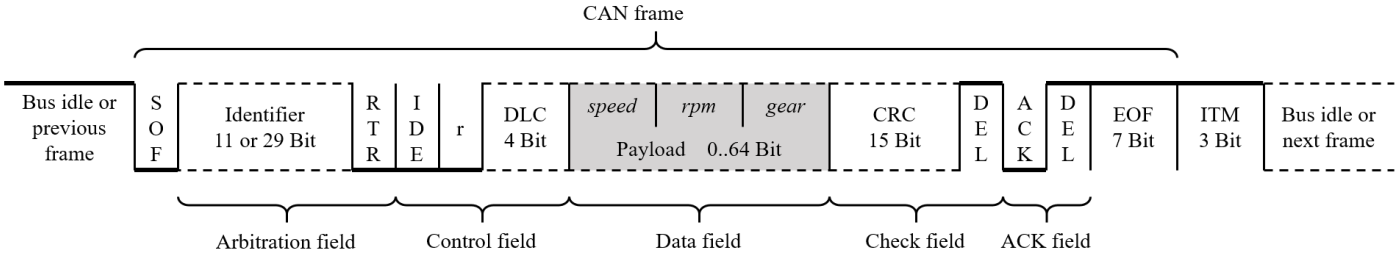

Fig. 1: CAN frame with exemplary payload

Nr	Sensor	Description
S-1	Formality	Correct message size, header and field size, field delimiters, checksum, etc.
S-2	Location	Message is allowed with respect to dedicated bus system
S-3	Range	Compliance of payload in terms of data range
S-4	Frequency	Timing behavior of messages is approved
S-5	Correlation	Correlation of messages on different bus systems adheres to specification
S-6	Protocol	Correct order, start-time, etc. of internal challenge-response protocols
S-7	Plausibility	Content of message payload is plausible, no infeasible correlation with previous values
S-8	Consistency	Data from redundant sources is consistent

Fig. 2: Sensor classes defined by Müter et al. (from [12])

Detection Sensor \ Criterion	Specification-Based	Number of Messages	Number of Bus Systems	Different Message Types	Payload-Inspection	Semantic-Based
Formality	*true*	1	1	*n.a.*	*false*	*false*
Location	*true*	1	1	*n.a.*	*false*	*false*
Range	*true*	1	1	*n.a.*	*true*	*false*
Frequency	*true*	*n*	1	*false*	*false*	*false*
Correlation	*true*	*n*	*n*	*true*	*false*	*false*
Protocol	*true*	*n*	*n*	*true*	*false*	*false*
Plausibility	*false*	*n*	1	*false*	*true*	*true*
Consistency	*false*	*n*	*n*	*true*	*true*	*true*

Fig. 3: Applicability of anomaly detection sensors (from [12])

Anomalies can occur on message level as well as on payload level. It is worth to note that all detection sensors, working on message level (*Payload-Inspection* set to *false* in Figure 3), are classified as specification-based. This type of checks can be implemented efficiently without using machine learning algorithms. It is not necessary to learn e.g. the period of a CAN message, if it is already statically defined in the communication matrix. But there are payload properties, which cannot be checked purely based on a given specification. One example is the time series of a signal. While absolute minimum and maximum values are usually defined, there is no explicit information about the normal temporal behavior of a signal given statically. However, this temporal behavior can be checked for anomalies by applying machine learning algorithms on the corresponding time series data as shown e.g. by Andreas Theissler [13], although his work focuses on offline analysis instead of real-time intrusion detection on ECUs.

IV. Hybrid Anomaly Detection System

The basic idea of the proposed system is to use specification-based anomaly detection and machine learning algorithms sequentially within the embedded software of an ECU. Figure 4 depicts the principle building blocks of the two-stage system. Because of not issuing false positives and due to the possibility to implement them efficiently, specification-based checks are applied in the first stage and are favored over machine learning algorithms. But there is one restriction: The checks shall be completely derivable from a communication matrix of standardized format. This first stage is further referred to as *static checks*. Müter et al. classified their first six sensor classes (S-1 to S-6) as specification-based [12], see Figure 3. Therefore, they are candidates for static checks. However, there are some differences regarding the underlying specifications. Some checks of the formality sensor class (S-1) only refer to a protocol specification and do not require OEM or vehicle specific information. As long as the protocol specification is standardized, this is no issue and a realization within the static checks is possible. Location (S-2) and range (S-3) sensors are derivable from a communication matrix, given that the value range of signals is defined. For the frequency sensor (S-4) the situation is different. Checks of this class can be derived for periodic messages, whereas the implementation of frequency sensors for purely event-driven messages is more challenging. For this purpose, advanced mechanisms like machine learning can be used, which e.g. determine the minimum and maximum period of such a message during their training phase. The required information to implement correlation sensors (S-5) can also be included in a communication matrix of a gateway ECU which bridges different communication networks. Protocol sensors (S-6) are related to formality sensors. As an example, in case of diagnostic requests and responses, a protocol sensor can check whether the response does formally belong to the preceding request based on the standardized

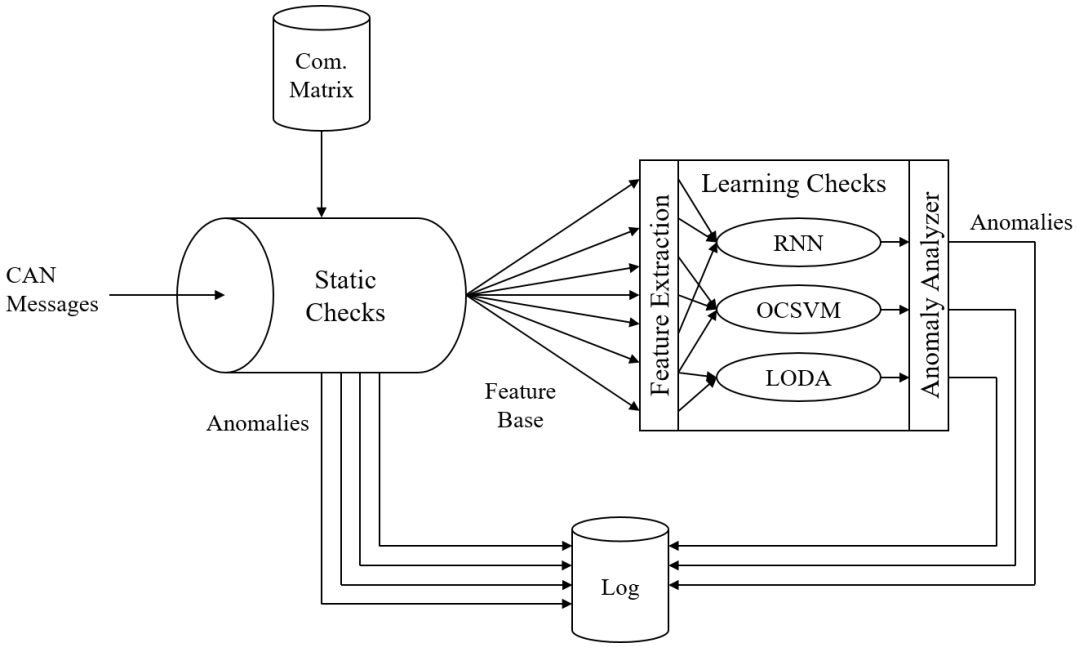

Fig. 4: Block diagram of the hybrid anomaly detection system for CAN

Unified Diagnostic Services (UDS) specification. However, some OEMs also use small proprietary protocols, e.g. on top of CAN communication, which could be checked in a similar way. The required information to implement these kind of protocol checks is mostly available in an OEM-specific and not standardized specifications. Therefore, such sensors are currently not realized as part of the static checks.

In the second stage of the system, *learning checks* extend the detection possibilities beyond the purely specification-based approach. To apply the corresponding machine learning algorithms, relevant information has to be extracted from CAN messages. The static checks only forward selected data elements like signal values to the learning checks, depicted in Figure 4 as *feature base*. The *feature extraction* block, pre-processes the data elements and generates features, which represent the input data for the following algorithms. Pre-processing can contain multiple aspects, like building time series, calculating derivations and normalization. Coming back to the sensors, defined by Müter et al. [12], the last two classes, namely plausibility (S-7) and consistency sensors (S-8), require semantic information about the transported signals, which is neither included in the protocol specification nor in the communication matrix. Therefore, static checks cannot be derived for these sensors. To realize consistency sensors, probably the evaluation of additional OEM-specific specifications is necessary to determine the signals between which the semantic consistency can be checked. In contrast, plausibility sensors focus on the temporal behavior of one communication signal. The single signals can be extracted from CAN messages by using the communication matrix and their temporal behavior can be examined using learning checks. This topic is further discussed in the second part of the paper at hand, starting with section V.

From another perspective, static checks are very well suited to detect point anomalies like signal values out of range (range sensor). In addition, simple collective anomalies can be detected efficiently using static checks, as long as they can be derived from the communication matrix, e.g. the period evaluation of cyclic messages (frequency sensor). Learning checks are required for advanced contextual and collective anomalies, which cannot be detected based on the communication matrix, e.g. an unnatural time series of a communication signal (plausibility sensor).

In their state of the art analysis, Chandola et al. [11] present different machine learning algorithms for anomaly detection. Each of them is well suited for different use cases. Therefore, the proposed system allows using a variety of algorithms, working on the same features. Figure 4 shows three examples: Replicator Neural Networks (RNN) [14], One-Class Support Vector Machines (OCSVM) [15] and Lightweight On-line Detector of Anomalies [16]. This list is by far not complete and can be extended easily. Each algorithm produces as output either a binary value ('normal'/'anomaly') or a so-called anomaly score, which represents the probability of an anomaly. These produced outputs are evaluated within the *anomaly analyzer* block, which e.g. checks an anomaly score for a defined threshold value before it securely stores the anomaly in a log. The design of the learning checks also allows for ensemble-based methods, as proposed in recent research e.g. by Andreas Theissler [17]. In an ensemble, multiple algorithms run in parallel, checking for the same anomalies and each of them producing its own output. In such a setup, the anomaly analyzer performs a voting between the different outputs and finally decides whether an anomaly is logged or not. This kind of post-processing is not necessary for static checks, since they do not work with probabilities. Therefore, they directly log the detected anomalies together with the corresponding boundary conditions (e.g. which CAN frame caused the anomaly) to enable an improved post-evaluation.

V. Anomaly Detection in Time Series Data

Data exchange between ECUs via CAN signals is vital for the majority of the implemented vehicle functions like engine control and electronic stability control. Besides the pre-defined signal properties like absolute minimum and maximum value the time series of a signal - further referred to as signal sequence - is one major characteristic. Especially signals, representing continuous physical values, are well suited for anomaly detection in their sequence. As an example, the speed of a vehicle is bound to physical constraints. It cannot change too rapidly and some signal sequences are very unlikely to occur, e.g. oscillation. These constraints are not pre-defined within the communication matrix and therefore cannot be observed by static checks. However, wrong signal sequences are collective anomalies, which are detectable by learning checks representing a plausibility sensor [12].

In a first step, a machine learning algorithm has to be selected, which is suited for real-time anomaly detection in signal sequences. There are several criteria which have to be considered for the decision. First, it is important whether the training is done on dedicated infrastructure like a workstation (offline) or on the target system itself (online). For most algorithms, the training is computationally expensive and therefore not applicable on ECUs whereas the classification task itself requires less processing power. On the other hand, online learning is required if the algorithm needs to adapt its behavior to changing system constraints during runtime. To account for this so-called concept drift [18], the training of the algorithm is continued while classifying data instances. Later decisions of the algorithm are influenced by the parameter adaption, which is usually performed after each processed data instance. However, many algorithms do not support the continuous adaption of parameters with every data instance, since they can only be trained with a complete data set.

The second important criteria is the selection of supervised or unsupervised learning, dependent on the available training data. Supervised learning requires the existence of labeled data, i.e. each data instance has to be annotated with the expected result - in this case *normal* or *anomaly*. Considering anomaly detection, this knowledge is used to learn a classifier, which shall be able to distinguish normal from anomalous data instances. However, to obtain labeled data for anomalous signal sequences in a large scale is difficult or even impossible because of the data collection and labeling effort. Additionally, training with anomalous data would result in a classifier, which is probably only able to detect known anomalies. This contradicts the idea of anomaly detection, which tries to find any kind of deviation from normal behavior. Due to these reasons, supervised learning is not further considered in the selection process of finding an appropriate algorithm. Unsupervised learning does not require labeled data and is mostly used for clustering techniques. These algorithms group data instances and thereby try to find the classes *normal* and *anomaly* automatically. On the other hand, most of these algorithms require a lot of data instances to be available during runtime in order to perform the grouping. Since the proposed anomaly detection system shall be implemented on an embedded ECU, considering the required resources - in terms of computing power and storage consumption - is important. Both is limited on an embedded device, compared to office computers or server systems. Following this constraint, the computational complexity of the selected algorithm must be low when executed on an ECU to detect anomalies.

Name	Time Complexity	Storage Complexity	Description
HS-Trees	$\mathcal{O}(t(h+l))$	$\mathcal{O}(t2^h)$	t: Number of parallel trees h: Maximum tree depth l: Sliding window length
LODA	$\mathcal{O}(k(d^{-\frac{1}{2}}+b))$	$\mathcal{O}(k(d^{-\frac{1}{2}}+b))$	k: Number of histograms d: Number of input features b: Number of bins

TABLE I: Comparison of the big O-notation between LODA and HS-Trees (Source [16][23]). The time complexity of HS-Trees and LODA includes the time to update the classifier.[1]

An unsupervised method, that is especially designed for anomaly detection, is Isolation Forest (iForest) [19]. The basic idea of iForest is that anomalous data instances can be isolated easier than normal data instances. The isolation is done by randomly selecting a feature and performing axis-parallel splits until all data instances are separated from each other. This process is repeated multiple times from scratch to construct diverse decisions trees. This can be seen on an conceptual level as the exploration of random local sub-spaces [20]. The anomaly score of a tested data instance is computed by checking the average depth in all trees. Data instances with small average tree depth have a higher anomaly score, because they need less splits to be separated [19]. iForest is an example for ensemble-based machine learning algorithms. This type of algorithm combines multiple weak classifiers and votes on the total outcome. Ensemble-based methods can handle a large number of input data instances with many features and improve speed by making parallel processing possible. Ensembles can be used within one algorithm as done by iForest but also between different algorithms as described at the end of section IV and in [17]. Different researchers published optimizations for iForest targeting different use cases [21][22]. The most notable approach for online learning are Half-Space-Trees (HS-Trees) [23]. This algorithm randomly splits the input data space until a predefined maximum tree depth is reached or there is just one data instance left in the corresponding branch of the tree. The procedure is repeated to build an ensemble of trees similar to the iForest algorithm. Each end point of a tree represents a remaining region, which is not split further. The value of the end point is the number of data instances in the corresponding region (mass). A data instance, which has to be classified, is traversed through all trees. Thereby, the mass of all traversed end points is recorded and their average is calculated. This average mass represents the anomaly score of the data instance. The lower the average mass, the higher is the probability for an anomaly.

Lightweight On-Line Detector of Anomalies (LODA) is an unsupervised anomaly detection algorithm especially designed for low time and storage complexity with the capability of

[1]The storage complexity of HS-Trees in the LODA-Paper [16] differs from the complexity given by the authors of HS-Trees [23]. The storage complexity given here is the complexity taken from the HS-Tree-Paper.

online learning. LODAs' concept of using random projections is similar to iForest and HS-Trees. The time complexity is not only deterministic, but it also can be sub-linear with respect to the number of input features, as shown in Table I. Another advantage is that there are no parameters to be set or tuned manually, but all parameters are optimized automatically. For other algorithms, especially neuronal networks, tuning those parameters can be a complicated and long process. LODA is also able to handle missing data values and it allows to figure out the feature(s) causing an anomaly. Even though normalizing values (see section VI-A) improves the detection rate, LODA can - in contrast to most other machine learning algorithms - cope with values, that are out of the normalized scale. The described HS-Trees for example can only use values that are bound between zero and one. This is a disadvantage when applied in real world applications because the range of values might change due to concept drift or there is no predefined range of values available but a range has to be estimated.
The author states, that in his tests LODA outperformed HS-Trees by factor seven to eight [16]. By benchmarking eight selected machine learning algorithms, Emmot et al. found that only iForest and LODA are able to scale to big data sets [24]. Both were able to generally outperform OCSVM [15] and Support Vector Data Description (SVDD) [25].
Considering the discussion above and the future goal of online learning on ECUs, LODA is a promising algorithm for the learning checks of the proposed anomaly detection system and is investigated in more detail.

VI. Lightweight On-line Detector of Anomalies

This section gives an overview about the working principles of LODA as proposed by Tomáš Pevný in 2016 [16]. The basis of the classifier are randomly projected histograms whose generation is described in the following paragraph.
In the first step a sparse random projection is applied on the input data instance. A random projection is constructed by setting up a vector of the same dimension as the number of input features and assigning element values from $\mathcal{N}(0,1)$, as recommended by the Johnson-Lindenstrauss lemma to approximately maintain L_2-Distance [26]. The sparsity of the vector is created by setting random elements to zero. The probability of setting an element to zero is $1 - \frac{1}{\sqrt{d}}$ as recommended by Li et al. [27], where d is the dimensionality of the input data. The reason for projecting data instances randomly instead of using single features to construct the histograms is that diverse sub-spaces improve the detection rate. This type of diversity is not only used in iForest but also in most other ensemble based algorithms like the supervised Random Forest [28][16].
In the second step an one dimensional histogram is constructed for each projection of the input data. Tests showed that two alternating equi-width histograms work best for LODA [16]. An equi-width histogram is constructed by dividing the value axis in bins (intervals) of equal width. For online learning, two alternating histograms are used. An incoming data instance is classified using the first histogram. Simultaneously, the second histogram is constructed including the new data instance. At the end of the classification the histograms are exchanged. The number of bins is especially important since it influences the time and storage complexity of the algorithm as seen in Table I. There are several methods for computing this number like Sturges' formula [29], Scotts' normal reference rule [30] or the Freedman-Diaconis rule [31]. The easiest and default rule for most statistical software is Sturges' formula [29] which calculates the number of bins b to $b = 1+\log_2 n$, where n is the number of initial data instances. However, for large data sets, Sturges' formula is too conservative. Therefore, the method by Birgé and Rozenholc [32][33] is used for LODA, which optimizes the penalized maximum likelihood L_n as shown in equation 1.

$$L_n(b) = \sum_{i=1}^{b} n_i \log\left(\frac{bn_i}{n}\right) \underbrace{-b + 1 - \{\log(b)\}^{2,5}}_{\substack{\text{discount factor,} \\ \text{sanctioning of high } b}} \tag{1}$$

Here, n_i is the number of data instances, that fall in the i-th bin. This method provides good results for high dimensional data and has an acceptable computational cost [33]. In the end, the anomaly score f of a data instance x can be calculated as follows:

$$f(x) = -\frac{1}{k}\sum_{i=1}^{k} \log \hat{p}_i(x^T w_i) = -\log\left(\prod_{i=1}^{k} \hat{p}_i(x^T w_i)\right)^{\frac{1}{k}} \tag{2}$$

$x^T w_i$ projects a data instance to a scalar value z_i. The joint probability $\hat{p}_i(z_i)$ is calculated as the number of data instances in the same bin as z_i, divided by the total number of considered data instances. This procedure is repeated for all k projections. Finally, the average joint probability value over all histogramms is calculated. In the following statement, Pevný describes the anomaly score of LODA, where a sample corresponds to a data instance.

Loda's output is proportional to the negative log-likelihood of the sample, which means that the less likely a sample is, the higher the anomaly value it receives. [16]

The maximum number of projections and associated histograms can either be defined manually or is also automatically optimized. By defining the number of histograms the necessary storage can be limited to a fixed size. To automatically find the minimum number of projections, the reduced variance $\hat{\sigma}_k$ after adding a histogram is measured. The variance is computed by the following equation, where f is the anomaly function described in equation 2, k is the number of projections and n the number of data instances.

$$\hat{\sigma}_k = \frac{1}{n}\sum_{i=1}^{n} |f_{k+1}(x_i) - f_k(x_i)| \tag{3}$$

With the given equation, a high number of projections will not be sanctioned since the variance decreases with each added projection. Therefore, the variance is normalized by the first variance $\hat{\sigma}_1$ and a new projection is only added if the normalized variance change is bigger than a predefined threshold τ. For LODA the recommended threshold τ is 0.01 which was used for all following experiments.

A. LODA for Time Series Data

In the proposed system, LODA is used as a learning check for observing the time series of a communication signal. Regarding the implementation, first the sequences of CAN signals have to be made available as input values for the machine learning algorithms of the anomaly detection system. Every time a CAN message is received, the static checks extract the single signals, which are then passed to the feature extraction block, please refer to Figure 4. This block builds up and manages the time series of those signals with a configurable count of history values. Internally, it implements a sliding window mechanism, which shifts the already available signal values as soon as a new value arrives. The oldest value is disposed.

The input values for the machine learning algorithms like LODA are not directly the CAN signal values but their normalized representation. A normalization around zero is performed by column-wise standardization $\left(\frac{x_i - mean(x)}{std(x)}\right)$. For later work there are several other online and offline normalization approaches [34] that might be able to improve the results and can for example utilize the minimum and maximum values of signals defined in the communication matrix.

The training itself is realized by a special program, which takes recorded CAN logs as input. Since all data is available at once, currently offline training in batch mode is used for simplicity reasons. Online learning is going to be the next step of our research.

VII. Evaluation

To evaluate the performance of the proposed system, a synthetic CAN signal is used. The signal sequence as well as the corresponding CAN messages are generated with CANoe from Vector Informatik, which is a simulation and analysis tool for automotive networks. A corresponding communication matrix is used to automatically generate the static checks including the signal extraction from CAN messages. The signal sequence itself is periodic and one period consists of approximately 850 samples emulating a physical value of a vehicle, e.g. speed. It is depicted in Figure 5 as *reference signal* and represents the normal behavior for the following considerations.

For the evaluation of the anomaly detection performance, an altered signal sequence was defined as well, containing five anomalies of different types. Anomalies are inserted starting from sample 1350. The *anomalous signal* as well as the highlighted anomalies are shown in the lower part of Figure 5.

1) 1350-1550 - limitation of the value range: This anomaly limits the minimum and maximum value of the signal, which is different from the pre-defined one.
2) 1900-2100 - value freeze: In this case, a value is kept for an unnatural long time before it resumes to the original signal sequence.
3) 2800-2950 - alternative signal sequence: In contrast to the two types discussed before, an alternative sequence is not an anomaly. Instead it shall be a first check of the generalization of the trained algorithm. Alternative signal sequences, which represent another form of realistic or normal behavior, shall not be detected as anomaly. Otherwise, many false positive alarms would be issued when implemented in a real ECU, since e.g. the actual sequence of speed values will never be completely identical to the sequences contained in the training data set.
4) 3650-3750 - peak: The peak anomaly refers to a spike in the signal sequence, which should not be present.
5) 4450-4600 - signal jump: Instead of having a continuous shape, the signal sequence includes an unrealistic high value step. This type of dramatic value change is impossible, e.g. the vehicle speed cannot instantly switch from a high value to zero.

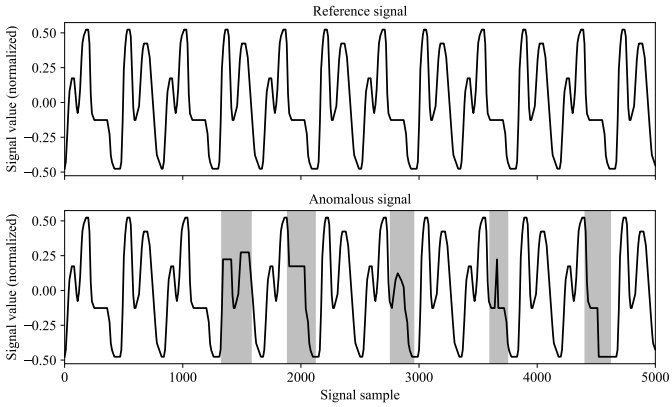

Fig. 5: CAN signal sequences - reference and anomalous signal

As discussed before, no paramters need to be tuned to execute LODA. All parameters are automatically derived and optimized as described in section VI. The only tuning point, not directly related to LODA, is the size of the sliding window, i.e. the number of history values of the signal sequence considered for anomaly detetion.

A. Amount of Training Data

The first evaluation concerns the amount of training data necessary for LODA to function properly. One period of the reference signal (sample 0-850 in Figure 5) was periodically repeated a specified number of times. Afterwards the anomaly score was determined as shown in equation 2 by classifying the anomalous signal.

As shown in Figure 6 the amount of normal training data has influence on the anomaly score but in all three cases, LODA detects the anomalies. In case only five reference signal periods are used for training (last row), the anomaly score is noisy, which makes thresholding difficult and which could lead to an increased false positive rate. However, even with very little training data a surprisingly good classification is possible. Noise and baseline of the anomaly score (visible on the left side of the third and forth row) decreases with more training data. Also, the difference in anomaly score between normal data and anomalies increases - comparable to a better signal-to-noise ratio.

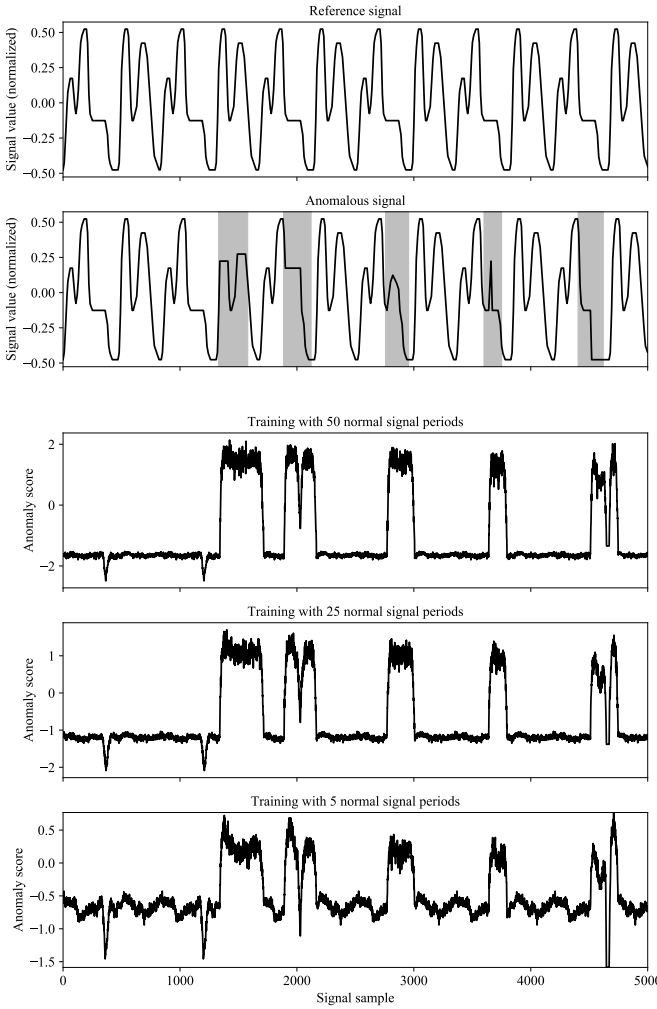

Fig. 6: Anomaly detection performance of LODA after a different number of reference signal periods for training. Please note the different scaling of the y-axis (anomaly score) and that the input values (reference and anomalous signal) are normalized as described in section IV.

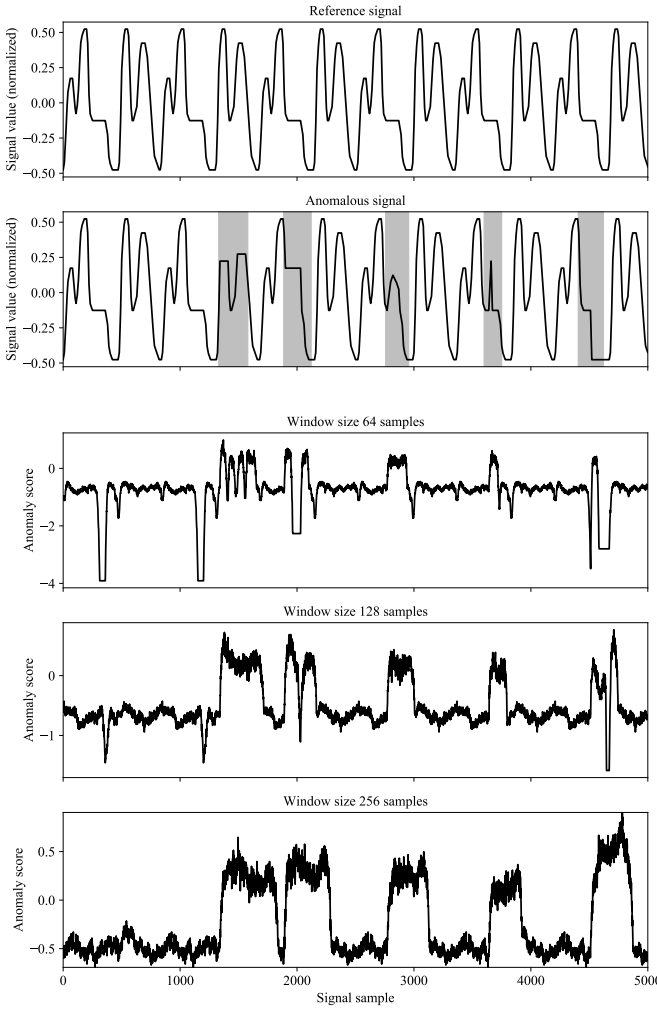

Fig. 7: Anomaly detection performance of LODA with a window size of 64, 128 and 256 signal samples (history values).

One issue in this evaluation is that the alternative signal sequence (sample 2800-2950) has a high anomaly score independent of the amount of training data or the window size described in the next sub-section. However, some concepts to overcome this issue are introduced in section VII-C.

B. Window Size

The second evaluation targets the influence of different window sizes (the number of history values of the signal sequence taken into account) on the resulting anomaly score. Figure 7 shows the results. Again the first two rows show the reference and anomalous signal. The next three rows depict the anomaly score calculated by LODA with different window sizes.

LODA was again able to detect all anomalies with any given window size. The larger the window size the more easily is the identification of anomalies in the anomaly score, which makes thresholding easier. But due to some associated disadvantages, the window size cannot be increased too much. First, a larger window size increases the relative noise on the anomaly score. Second, it gets harder to pinpoint the exact moment of an anomaly since the anomaly score is influenced longer. Third, the required computing power and storage consumption increases - even though moderately compared to other machine learning algorithms. A larger window size is also relevant for the startup time of the system, because it has to be waited for the first n samples until a classification can be performed. For example, with a window size of 256 samples and a typical CAN message period of 10ms the startup time is going to be about 2.5s. Choosing the right window size is an issue that needs to be addressed in future work. For the other evaluations within this paper a fixed window size of 128 samples is used.

C. Detection of the Alternative Signal Sequence

In the previous evaluations, the alternative signal sequence, described at the beginning of this section, is classified as anomaly even though it represents a valid sequence and should not be detected. Up to now, the training data consists of

multiple identical reference signal periods. To get a first indication whether more diverse training data improves the situation, the reference signal is modified. For this evaluation, it consists of original reference signal periods alternating with signal periods including the alternative sequence, see first row of Figure 8. For real world applications also diverse training data is used and it is likely that normal signal sequences are contained several times in slightly different forms.

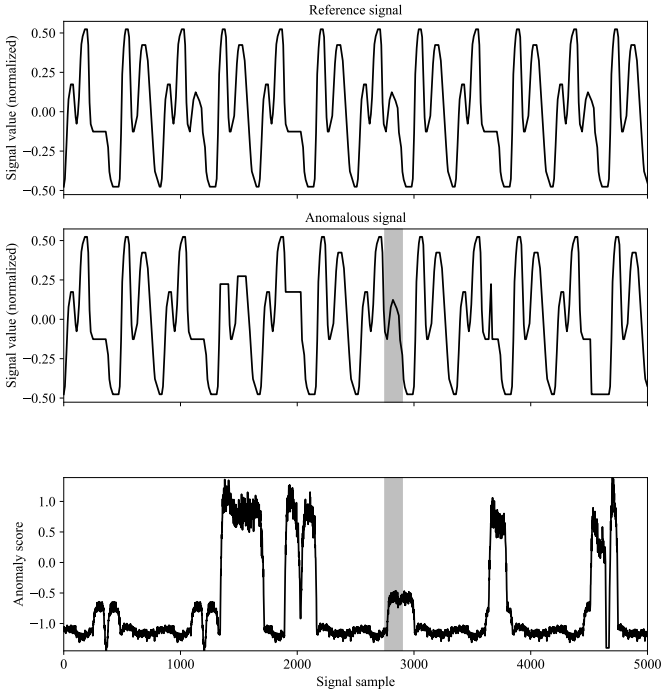

Fig. 8: Anomaly detection performance of LODA with the alternative signal sequence injected in the training data.

As highlighted in Figure 8, the anomaly score during the alternative signal sequence is now similar to the baseline and therefore classified as normal data. The other anomalies are still detected. However, the anomaly score for parts of the original signal sequence is now increased (sample 300-500 and sample 1100-1300). This can be explained by the fact that a constant signal value over several samples is present less often in the modified training data. Overall, diverse training data is important for LODA to avoid overfitting. This fact has to be investigated in more detail in future using real signal sequences e.g. extracted from CAN logs of test drives.

VIII. Conclusion and Future Work

This paper motivated the need for intrusion detection systems in automotive ECUs. Today, anomaly detection, which comes along with IDS, is mostly used within information technology (IT) infrastructure like servers and networking equipment. Since automotive and embedded networks are far more static than IT infrastructure, a different approach is proposed. Instead of using a solution, purely based on machine learning or other complex algorithms, we favor a hybrid anomaly detection system. In a first stage, the system works specification-based. Especially in the automotive industry, there exist semi-formal network specifications which are perfectly suited to implement an efficient and effective IDS. However, the specifications do not contain information about the temporal behavior of communication signals. For the advanced observation of this temporal behavior machine learning algorithms can be used.

In the second part of this paper the maschine learning algorithm LODA was investigated in more detail by focusing on anomaly detection in time series data. It was shown that LODA can deliver good results in this use case. Also the low complexity of the algorithm is a plus, especially when it comes to online learning. However, only testing with real data in a real environment will show whether the performance is good enough to be applicable in an ECU. In future, all components of LODA have to be examined more thoroughly. Also, different thresholding techniques to convert the anomaly score into the classes *normal* and *anomaly* have to be investigated.

Since the evaluation is up to now based on synthesized communication signals, the next step is to perform similar tests with real data. As a starting point, vehicle signals, collected via the standardized On-Board Diagnostics II port, can be used. However, in a final step, real bus logs should be used to do a performance evaluation under the same conditions as if the anomaly detection system would run in a real ECU.

A second field of work is the extension of the proposed system to support additional bus and networking systems. As Ethernet is currently becoming an important in-vehicle networking technology, the extension by static checks for Ethernet would be another next step. Since the learning checks work on communication signals, they are independent from the underlying networking or bus system. Therefore, the investigated LODA algorithm is re-usable in an extended hybrid anomaly detection system.

In future, also a more detailed classification of detected anomalies is required to enable appropriate countermeasures. However, this is a different topic of research and not addressed by this work.

References

[1] C. Miller and C. Valasek, "A survey of remote automotive attack surfaces," *Black Hat USA*, vol. 2014, 2014.

[2] ——, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, 2015.

[3] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno *et al.*, "Comprehensive experimental analyses of automotive attack surfaces," in *USENIX Security Symposium*, 2011.

[4] H. Onishi, "Paradigm change of vehicle cyber security," in *4th International Conference on Cyber Conflict (CYCON), 2012*, C. Czosseck, Ed. Piscataway, NJ: IEEE, 2012.

[5] AUTO-ISAC, *Automotive Cybersecurity Best Practices: Executive Summary*, 2016th ed. AUTO-ISAC, 2016.

[6] D. A. Brown, G. Cooper, I. Gilvarry, D. Grawrock, A. Rajan, A. Tatourian, R. Venugopalan, C. Vishik, D. Wheeler, M. Zhao, D. Clare, S. Fry, H. Handschuh, H. Patil, C. Poulin, A. Wasicek, and R. Wood, *Automotive Security Best Practices: Recommendations for security and privacy in the era of the next-generation car. White Paper*. McAfee, Inc., 2015.

[7] T. van Roermund and A. Birnie, *A multi-layer vehicle security framework: Whitepaper*, may 2016 ed. NXP B.V., 2016.

[8] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive can networks—practical examples and selected short-term countermeasures," *Reliability Engineering & System Safety*, vol. 96, no. 1, pp. 11–25, 2011.

[9] I. Studnia, V. Nicomette, E. Alata, Y. Deswarte, M. Kaâniche, and Y. Laarouchi, "Security of embedded automotive networks: state of the art and a research proposal," in *SAFECOMP 2013 - Workshop CARS (2nd Workshop on Critical Automotive applications : Robustness & Safety) of the 32nd International Conference on Computer Safety, Reliability and Security*, M. Roy, Ed., Toulouse, France, 2013.

[10] D. M. Hawkins, *Identification of Outliers*, ser. Monographs on Applied Probability and Statistics. Dordrecht: Springer, 1980.

[11] V. Chandola, A. Banerjee, and V. Kumar, "Anomaly detection: A survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 1–58, 2009.

[12] M. Müter, A. Groll, and F. C. Freiling, "A structured approach to anomaly detection for in-vehicle networks," in *Sixth International Conference on Information Assurance and Security (IAS), 2010*. Piscataway, NJ: IEEE, 2010, pp. 92–98.

[13] A. Theissler, "Anomaly detection in recordings from in-vehicle networks," in *Big Data Applications and Principles: Proceedings*, A. Mozo and Hernando, Antonio, Gómez (Eds.), Sandra, Eds., 2014, pp. 23–38.

[14] S. Hawkins, H. He, G. Williams, and R. Baxter, "Outlier detection using replicator neural networks," in *Data Warehousing and Knowledge Discovery*, ser. Lecture Notes in Computer Science, Y. Kambayashi, M. Arikawa, and W. Winiwarter, Eds. Berlin, Heidelberg: Springer-Verlag Berlin Heidelberg, 2002.

[15] B. Schölkopf, R. Williamson, A. Smola, J. Shawe-Taylor, and J. Platt, "Support vector method for novelty detection," in *Advances in Neural Information Processing Systems 12*. Cambridge, MA, USA: MIT Press, 2000, pp. 582–588.

[16] T. Pevný, "Loda: Lightweight on-line detector of anomalies," *Machine Learning*, vol. 102, no. 2, pp. 275–304, 2016.

[17] A. Theissler, "Detecting known and unknown faults in automotive systems using ensemble-based anomaly detection," *Knowledge-Based Systems*, vol. 123, pp. 163–173, 2017.

[18] J. Gama, "A survey on learning from data streams: Current and future trends," *Progress in Artificial Intelligence*, vol. 1, no. 1, pp. 45–55, 2012.

[19] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Eighth IEEE International Conference on Data Mining, 2008*. Piscataway, NJ and Piscataway, NJ: IEEE, 2008, pp. 413–422.

[20] C. C. Aggarwal, *Outlier Analysis*, 2nd ed. Cham and s.l.: Springer International Publishing, 2017.

[21] Z. Ding and M. Fei, "An anomaly detection approach based on isolation forest algorithm for streaming data using sliding window," *IFAC Proceedings Volumes*, vol. 46, no. 20, pp. 12–17, 2013.

[22] L. Sun, S. Versteeg, S. Boztas, and A. Rao, "Detecting anomalous user behavior using an extended isolation forest algorithm: An enterprise case study," *arXiv preprint arXiv:1609.06676*, 2016.

[23] S. C. Tan, K. M. Ting, and T. F. Liu, "Fast anomaly detection for streaming data," in *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, T. Walsh, Ed. Menlo Park, California: AAAI Press, 2011, pp. 1511–1516.

[24] A. Emmott, S. Das, T. Dietterich, A. Fern, and W.-K. Wong, "A meta-analysis of the anomaly detection problem," *arXiv preprint arXiv:1503.01158*, 2015.

[25] D. M. Tax and R. P. Duin, "Support vector data description," *Machine Learning*, vol. 54, no. 1, pp. 45–66, 2004.

[26] W. B. Johnson and J. Lindenstrauss, "Extensions of lipschitz mappings into a hilbert space," in *Conference on Modern Analysis and Probability*, ser. Contemporary Mathematics, R. Beals, A. Beck, A. Bellow, and A. Hajian, Eds. Providence, Rhode Island: American Mathematical Society, 1984, vol. 26, pp. 189–206.

[27] P. Li, T. J. Hastie, and K. W. Church, "Very sparse random projections," in *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '06*, L. Ungar, M. Craven, D. Gunopulos, and T. Eliassi-Rad, Eds. New York, New York, USA: ACM Press, 2006, pp. 287–296.

[28] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.

[29] H. A. Sturges, "The choice of a class interval," *Journal of the American Statistical Association*, vol. 21, no. 153, pp. 65–66, 1926.

[30] D. W. Scott, "On optimal and data-based histograms," *Biometrika*, vol. 66, no. 3, pp. 605–610, 1979.

[31] D. Freedman and P. Diaconis, "On the histogram as a density estimator: L 2 theory," *Z. Wahrscheinlichkeitstheorie verw Gebiete (Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete)*, vol. 57, no. 4, pp. 453–476, 1981.

[32] L. Birgé and Y. Rozenholc, "How many bins should be put in a regular histogram," *ESAIM: Probability and Statistics*, vol. 10, pp. 24–45, 2006.

[33] L. Davies, U. Gather, D. Nordman, and H. Weinert, "A comparison of automatic histogram constructions," *ESAIM: Probability and Statistics*, vol. 13, pp. 181–196, 2009.

[34] I. Saarinen, "Adaptive real-time anomaly detection for multi-dimensional streaming data," Master's Thesis, Aalto University, Helsinki, 2017-02-22.