Anomaly-based intrusion detection for automotive networks

ANOMALY-BASED INTRUSION DETECTION FOR AUTOMOTIVE NETWORKS

BY

MIKHAIL SMOLIN

MASTER'S THESIS

Submitted in partial fulfillment of the
requirements for the degree of Master of Science
in Computer Science and Media

Department of Computer Science
Stuttgart Media University

Stuttgart, Germany

Abstract

For a long time, vehicle safety was understood to be essentially only the seat belt, a reasonably large crumple zone, and airbags. Traditionally, car manufacturers had to ensure the safety of drivers and passengers through devices which used to protect the occupants against physical damage, caused by accidents. In the era of connected cars, safety and security may also be impaired that people deliberately launch cyber attacks on vehicles, which is otherwise only known from the Internet. Recently, it has been shown that hackers can access a car's network and take remote control [1–5]. Nowadays, modern cars have full access to the Internet, using network communication to other vehicles and interact with the road infrastructure in accordance with local traffic conditions. Advances in technology, connected and autonomous vehicles create several attack surfaces for potential cyber attacks [6], [7]. On that account, information security comes to an increasingly higher significance within the automotive sector. Conventional network security strategies, such as firewalls and gateways can prevent an unauthorized external access to the network. When it comes to automotive cyber security, threats are, however, more subtle. The potential security risks are currently changing significantly, as the internetworking of physical devices intersects with the automotive industry where information security meets functional security. This risk is growing due to the increasing use of standardized components and open interfaces which lead a possible hacker to new challenges. A central approach to mitigate these vulnerabilities is the permanent observation and analysis of network data and asserting their legitimacy. The main objective is to detect possible intrusions by examining patterns in the observed data that deviate from the expected behavior. In this thesis, the applicability towards the problem of detecting anomaly-based intrusions in an automobile network by means of a self-learning mechanism is proposed and evaluated. Using in-vehicle network recordings of a modern sedan, a neural network is trained in order to detect simulated attacks as anomalous behavior deviating from the learned vehicle state.

Zusammenfassung

Lange Zeit verstand man unter Fahrzeugsicherheit im Wesentlichen den Dreipunktgurt, eine relativ große Knautschzone sowie Airbags. Um die Sicherheit von Autofahrern und Passagieren gewährleisten zu können, mussten Automobilhersteller für gewöhnlich Vorrichtungen anbringen, um so körperliche Schäden vorzubeugen, die bei Verkehrsunfällen entstehen. In der Connected-Car-Ära können zusätzlich Cyberangriffe die Fahrzeugsicherheit beeinträchtigen. Diese sonst nur aus dem Internet bekannten Attacken können gegenwärtig bewusst auf Autos verübt werden. Dass sich Hacker Zugriff in das Automobilnetzwerk verschaffen und somit ein Fahrzeug fernsteuern können, wurde jüngst bewiesen [1–5]. Heutzutage haben moderne Autos Internetzugang, sie vernetzen sich mit anderen Fahrzeugen und können mit ihrer Umwelt interagieren. Der technologische Fortschritt, vernetzte und selbstfahrende Fahrzeuge bieten Hackern zahlreiche Angriffsflächen für mögliche Cyberattacken [6], [7]. Aus diesem Grund wird der Cybersicherheit in der Automobilbranche eine besonders große Bedeutung zugeschrieben. Konventionelle Strategien aus der IT-Sicherheit, wie beispielsweise Firewalls und Gateways, können einen unautorisierten Systemzugriff verhindern. Im Hinblick auf den Automobilbereich, sind mögliche Bedrohungen allerdings subtiler. Aufgrund der Überschneidung von vernetzten Geräten mit der Automobilindustrie, vor allem dort, wo IT-Sicherheit auf funktionale Sicherheit trifft, ist das Angriffsrisiko nicht nur im stetigen Wandel, sondern steigt zudem durch den verstärkten Einsatz standardisierter Komponenten sowie offener Schnittstellen. Dadurch wird die Automobilbranche vor neue Herausforderungen gestellt. Ein Ansatz, um Autos vor Cyberangriffen zu schützen, besteht in der aktiven Überwachung des Fahrzeugnetzwerks, um Angriffe als Abweichung vom validen Überwachungszustand zu erkennen. Im Rahmen dieser Abschlussarbeit wird die Anwendbarkeit einer Anomalie-basierten Angriffserkennung auf Fahrzeugnetzwerke, im Sinne eines selbstlernenden Verfahrens, vorgestellt und evaluiert. Unter Verwendung aufgezeichneter Fahrzeug-Bus-Daten, wird ein neuronales Netz trainiert, um simulierte Angriffe auf das Fahrzeugnetzwerk zu erkennen.

Contents

Glossary

ABS Anti-lock Braking System. Prevents the wheels from locking during emergency braking and avoids uncontrolled vehicle skidding. 10

ACK Acknowledge Field. Confirmation field used to receipt a correct incoming CAN frame. 11

Actuator Component of machines that converts the ECU's electrical signal into mechanical control movement. 8

AI Artificial Intelligence. A branch of computer science where computers perform operations based on learning and decision making. 27

ANN Artificial Neural Network. A self-learning system which is designed after the human brain to recognize patterns. 5, 27, 30, 36, 37, 38, 41, 43, 44, 57, 61, 68, 69

ARMA Autoregressive-Moving Average. Linear, discrete models of predictive processes which are used for statistical analysis. 29

CAN Controller Area Network. Technology used in automobiles to link various electrical devices. 2, 9, 10, 11, 12, 22, 23, 27, 46, 48, 51, 53, 54, 56, 57, 58, 59, 60, 61, 62, 63, 66, 67, 68, 69, 70

CRC Cyclic Redundancy Check. Safety field used for transfer fault detection. 11

DFA Deterministic Finite Automaton. State machine that changes among the possible state input, in which it is currently located, into a uniquely determined state condition. 14

DNN Deep Neural Network. Deep learning is the name used for stacked neural networks; that is, networks composed of several layers of nodes. 5

ECU Electronic Control Unit. Controls one or more electrical subsystems in a vehicle's network. 1, 2, 3, 7, 8, 10, 11, 53, 66

ESC Electronic Stability Control. Electronically controlled driving assistance system, which counteracts by braking individual wheels. 10

FlexRay Scalable and flexible high-speed communication protocol with safety and security related features in mind. 10, 70

FSM Finite State Machine. Mathematical model of technical/mechanical machinery composed of finite states, transitions and actions. FSM process character strings and generate output. 14

ID Message Identifier. Defines the priority of a CAN message. Smaller IDs have a higher priority on the CAN. 11, 56, 58, 60, 66, 67

IDPS Intrusion Detection and Prevention Systems. Focused on identifying possible attacks, monitoring incidents and attempting to stop them. 31, 48, 51

IDS Intrusion Detection System. Software or device to detect attacks that are directed against a computer system or computer network. 3, 27, 31, 32, 69, 70

IoT Internet of Things. Integration of computing capacity and communication ability in devices, creating a network of distributed, intelligent systems. 8

IPS Intrusion Prevention System. Unlike an IDS, the IPS has no monitoring function, but directly controls the traffic to fend off attacks. 31, 48

k-NN k-Nearest Neighbors. A classification algorithm in which a class assignment is made in consideration of its neighbors. 28

LIN Local Interconnect Network. Low cost communication for intelligent sensors and actors where CAN's versatility is not required. 10, 70

MLP Multi-layer Perceptron. A neural network with one or more layers between input and output layer. 41, 42

MOST Media Oriented Systems Transport. Infotainment field bus which is used to transport audio, video, voice and data signals. 10, 70

NFA Non-Deterministic Finite Automaton. Unlike the DFA the possibilities are not unique, therefore, the automaton is not predefined what transition is to be chosen. 14

Node In the CAN topology each equally eligible participant is called a node. 10

OSI Open Systems Interconnection. Model which describes the communication requirements that must be met for different network components. 10

PCA Principle Component Analysis. Statistical technique to identify hidden patterns and their classification in data recordings. 21, 27

V2I Vehicle-to-Infrastructure. Wireless communication concept to realize data transfer between automobiles and infrastructural facilities. 1

V2V Vehicle-to-Vehicle. Mobile communication technology that allows automobiles to directly communicate with each other. 1

List of Figures

List of Tables

Part I

Fundamentals

Chapter 1

Introduction

This chapter is intended to give the reader an insight into the topic, and to provide an overview of the thesis at hand, summarizing its motivation, approach and the problem statement completed by an outline.

Cyber **attacks** targeting connected vehicles can endanger driver and passenger safety. As more and more cars are getting connected to the Internet it is imperative to deal with those risks before efforts in cyber crime will become vital.

With integrating embedded devices, so-called electronic control unit (ECU), most recently, a significant progress has been made in automotive systems. Mainly for controlling an electrical system, an ECU is not merely used for monitoring vehicular parameters but also for enhancing its energy efficiency and reducing a car's noise, vibration and the emission of carbon dioxide [8]. A substantial amount of embedded devices is used to monitor and control electro-mechanical systems. Examples of such embedded systems can be seen in home automation and modern cars. In recent times, location-based services such as vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) depend on computing devices to accomplish intra-vehicular communication [9] as well as inter-vehicular communication [10], [11]. These expansions of the domain of connected cars give rise to a number of security concerns.

1.1 Motivation and problem statement

Nowadays embedded (distributed) systems are used in several applications with wide spread and acceptance. Moreover, they become even more important due to the fact that embedded software in ECUs continues to increase in domains, complexity, and sophistication [12]. In addition, skepticism regarding software in everyday life applications, even when human existence is at stake, is diminishing or rather moving out of people's focus. Furthermore, steer-by-wire techniques are used in aviation for several years now and the emerging change in automobile motorization, i.e. the transformation from fuel driven to electrically operated engines, can be seen as a chance to establish them in the automotive field. When software keeps on conquering new and more responsible operational areas - and even if not - it is worth thinking about new possibilities of recognizing anomalous behavior in the software flow without delay, and with low effort in preparation, action and maintenance. The protection of vehicular systems along with their versatile interconnection and stored data against cyber attacks such as manipulation and unauthorized access is therefore of increasing relevance.

Data Source

As a basis of the practical part approach of this work, traces (i.e. a documented set of all messages sent within the automobile's network system during a defined time period) are used, recorded in a moving car for the time been driven without apparent malfunction of any of the vehicle's components. The system is based on a controller area network (CAN) structure (cf. 2.2.2) and all information necessary to understand the data is given. In order to be able to detect abnormalities in the CAN data stream or to evaluate the interaction between different ECUs in a vehicle's subsystem, the data transfer on the vehicular network and in some cases internal variables of ECUs are being recorded by data acquisition systems (in-vehicle data-logger) during the test drives. Source of the message-protocol is a middle class sedan. The parts responsible for the communication are the same as in an up-to-date automobile of a major manufacturer. Although the underlying CAN [13] technology is a de facto standard for serial communication, models cannot be derived without uncertainties. This is understood as a chance to test new approaches for model-building and safety-related functionalities. The longest protocols include more than 1.3 Million messages and were recorded while driving for approximately 80 minutes. The long term objective is to use the derived models for safety- and diagnosis-functions in current and future vehicles.

1.1.1 Motivation

As mentioned above, distributed systems are getting progressively important in many areas and as a result, there are a lot of use case scenarios for the concept and practical implementation for this thesis. The starting point is the aim to develop (parts of) an approach which is capable of recognizing an intrusion as an anomaly (q.v. 1.1.3) when observing the exchange of CAN messages between the participants of a bus system in a modern automobile. Since the CAN bus is a very common network [14] in the field of intrusion detection for vehicular networks, it is used for the practical parts of the work, i.e. the experiments and the evaluation. Possible ways of using the developed system can be split into two parts: The learning can take place either during a test drive with live broadcasting network messages, i.e. live learning, or on the basis of a recorded trace (preprocessed learning). Since it has to be guaranteed that the anomaly detection system is usable for other automotive network protocols than the CAN bus, the universality thought has to be kept in mind during all considerations. In this regard, it is an explicit objective to provide a concept and prototypical implementation applicable to other commonly used protocols next to the CAN which can be observed.

1.1.2 Anomaly detection: The state of the art

Since it is crucial to ensure correct behavior, methods used for anomaly detection in reliable systems rely on very accurate preceding steps and require a lot of working power. A typical straightforward approach of recognizing anomalies in a computational system is the observation and analysis of variables and parameters regarding logic and other contradictions. For example, the brake of a car should not or could not be activated while it was accelerating. This approach becomes significantly more complicated in case of a distributed system as the number of possible interactions increases rapidly with growing complexity. Other methods are known from the field of software testing. In this scientific surrounding, vast models are usually derived from specification or by observing variables or invariants. In [15–18] an overview of promising techniques is given. Chapter 3 provides more detailed descriptions of several methods. The basic idea of the task in this thesis is the consideration that it should be possible to convert the problem of detecting abnormal states in a vehicle into a data analytic problem and, furthermore, to minimize the effort that is necessary to build a model, when a positive behavior is learned. Regarding this, an approach is proposed in [19].

1.1.3 Problem statement

According to [20] most security-related attacks are either input- or output-based. In order to protect an in-vehicle network against intruders, two main security approaches can be used, cryptography and intrusion detection systems (IDS). The first-mentioned has the ability to protect the network from external attacks by ensuring the authentication and data integrity. On the downside, cryptography cannot prevent the threats caused by internal attacks, i.e. intruders within the automobile's network. The later named mechanism allows the detection of an activity deviating from the normal behavior within the network by analyzing an ECU and triggering an alarm when this ECU exhibits an anomalous behavior. The short word descriptions below (on the basis of the keywords contained in the title) are intended to give a first quick overview of the aim and the background of this thesis. In the following subsection, the problem that will be solved in this work is specified shortly. A more detailed definition of the stated problem is presented and discussed in chapter 5.

Definition 1.1.1. Anomaly: In the following chapters, this term denotes a behavior deviating from the standard (the benchmark has to be stated by definition).

Definition 1.1.2. Intrusion: An intrusion refers to gaining unauthorized access or to compromise a (computer) system by breaking the security of such a system or causing it to enter into an insecure state.

Definition 1.1.3. Detection: Detection means recognizing whether a specified event (e.g. an anomaly) takes place at a certain moment.

Definition 1.1.4. Network: A network should be understood as a distributed system, i.e. a structure with several participants and active exchange of observable messages.

Problem specification

Especially in automotive applications, the impact on one's personal life might be significant in case of a fault causing the car being immovable. In many cases, purely software-related malfunction causes the whole system to fail. While faults in hardware can be recognized and located with less effort, complex software and software-systems tend to develop unspecified behavior, especially in scenarios which haven't been tested before. It is widely accepted that

testing all possible state combinations and execution ways is - if even possible - not a convenient way of analyzing software [21], [22]. This thesis has to show a possibility of observing a distributed system and assessing if it works well. If there is a malfunction, the absence of hardware faults suggests the presence of a software fault that may lead to the system's failing. A priori knowledge regarding the system enables the interpretation of the traffic on the bus. One precondition is the existence of data traces representing drives without anomalies. As a first step of detecting possible intruders, anomalies have to be recognized. The existence of an anomaly implies the existence of a fault. To fulfill the main task of detecting anomalies in a stream of automobile network messages, three subtasks were identified during the work on this thesis: system modeling, complexity reduction and model fragmentation. As a fourth subtask the performance and quality of the concept have to be tested as well as evaluated, i.e. measured and judged. Additionally, evaluation and verification of the learned models by a metric (to be understood similar to a performance rating) have to be enabled.

1.2 Self-learning of decomposed automata

In order to solve the problem of representing a complex real system it is proposed to enable the self-learning of decomposed automata models. The principle aim is to find an abstracted data representation, which provides system access when an actual stream has to be verified. The presented solution is based on an adapted and refined method using a deep neural network (DNN) [23–25]. A basic multi-class anomaly detection technique using an artificial neural network (ANN) operates in two steps. First, an ANN is trained on the normal training data to learn the different normal classes. Second, each test instance is provided as an input to the ANN. If the network accepts the input it is normal or otherwise, it is an anomaly. The proposed approach is inspired by a set of algorithms with the main goal of modeling abstractions in data by incorporating a number of processing layers along with complex structures or otherwise, composed of a set of non-linear transformations. The feature vectors building the DNN structure are trained with probability-based parameters and have to be extracted from the trace. Since the message volume in distributed systems can be massive, the data is preprocessed so that several small automata can be learned. For this reason, a clustering of the individual data set is necessary, grouping events that exhibit related behavior.

1.3 Outline

This thesis is structured straightforward. Each chapter starts with a brief outline, allowing the reader to easily follow the chapter's content. In chapter 2 automotive systems and automata theory are explained and concluded. Elemental definitions and assumptions needed for the following chapters are provided and sorted likewise as the subsections as can be seen in this introduction. State-of-the-art solutions in related problems are described and explained in chapter 3 upfront. Chapter 4 discusses methods with regards to machine learning necessary for the following paragraphs. Subsequently, in chapter 5 the problem statement is described in more detail and stated formally. In chapter 6, the developed system is presented including its effectiveness. The results and possible future works in the related field are drawn and discussed in chapter 7 which completes this work.

Chapter 2

Automotive systems and automata

The subsequent chapter introduces necessary fundamentals and term definitions with respect to automotive systems together with mathematical background to ensure a convenient reading of this thesis work.

130 **years** after the invention of the first petrol-powered automobile, modern vehicles have turned from mechanical machines into complex systems of embedded software and electronics. This is equally applicable to industrial and agricultural engines [26].

Connected vehicles, whether it comes to cars, trucks, buses or construction machinery, use various wireless technologies [27]. With regard to this, a new set of on-board features and value-added services offer traffic jam warnings, reduction of fuel consumption or increasing performance. As a consequence of the shift from mechanically operated to software-driven vehicles, and the increasing connectivity, there is a high potential for security threats in modern automobiles [28]. A possible scenario would be an attacker injecting messages into the car's networks, directly or indirectly controlling the desired ECU with regard to performing various physical control operations. Another practicable situation might be the integration of compromised subsystems into the vehicle. Compounding this, attack surfaces are rapidly growing as more sophisticated services and features are integrated into modern vehicles [29].

2.1 Nomenclature

With consideration of easy comprehensibility, the used nomenclature in this thesis is introduced in the following section. Scalar variables and parameters are written in italics, for instance x, and matrices are named with uppercase letters which are printed in boldface (e.g. $\mathbf{X}$). To mark a vector, a standard arrow is used (e.g. $\vec{x}$), while constants will be abbreviated to a lowercase Greek letter such as α. Lowercase Latin letters denote a function (e.g. x). If a letter represents a tuple, it is printed in boldface and lowercase (e.g. $\mathbf{x}$). Finally, calligraphic uppercase letters specify value sets (e.g. $\mathcal{X}$). An overview of the definitions is given in table 2.1. All these rules are not valid for indices as they are always printed in lowercase italics.

Category	**Symbol**
Constants	α
Functions	x
Matrices	$\mathbf{X}$
Sets	$\mathcal{X}$
Tuples	$\mathbf{x}$
Variables/Parameters	x
Vectors	$\vec{x}$

Table 2.1: Typeface and notation of used symbols.

2.2 Automotive systems

Nowadays, the majority of automotive innovations is achieved by the means of software and electronics. Following a moderate but steady annual growth between 6 percent [30], [31] and 9 percent [32], the market for vehicle electronics is expected to have a yearly global volume of approximately 280 billion euros by the year 2020 [33]. As the internet of things IoT links automobiles by performing safety-critical functions without the need for human intervention, the growing technological progress in the automobile space is becoming more central to the act of mobility. As connected and driverless cars emerge, technologies such as Internet connectivity will become less an accessory and more a critical component. Connected vehicles and associated services will play an increasingly greater role, moreover, the vast number of software operating modern cars raises the likelihood of threats and vulnerabilities [34], [35].

2.2.1 E/E architecture

Modern cars contain up to 80 ECUs that perform various tasks over several bus systems [36], [37]. A typical job would be an ECU controlling the vehicle's engine. The main function is to get information from a multitude of sensors. This input is used to calculate control values and run certain actuators to enforce the actions determined by the modules. Several ECUs need to exchange data among themselves, share values and verify with each other during the vehicle's normal operation, resulting in a complex network of hard- and software subsystems [38] distributed by a diversity of suppliers [39]. The electric and electronic parts together with the corresponding network inside a modern executive car are shown in figure 2.1.

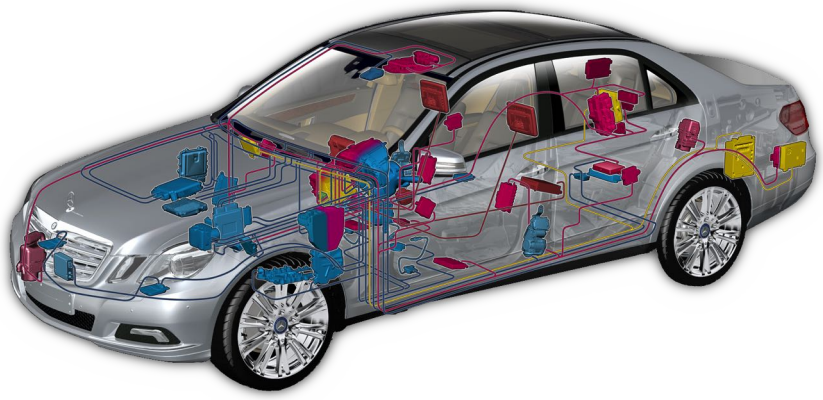


Figure 2.1: Electronic components and in-vehicle network in a modern executive car [40].

2.2.2 Controller area network

With the developments taking place in the electronics and semiconductor industry, mechanical in-vehicle systems were being replaced by more robust systems with improved performance. Providing a mechanism which is incorporated in the hardware and the software, the CAN is a common vehicular bus system used in European petrol and diesel vehicles since 2001 and 2004 respectively [41]. It is also adopted in other safety-related fields such as industrial automation and vehicle engineering in addition to aviation technology. The main functionality is to connect all the electronic components in a car. Before CAN, in-vehicle

devices were connected using point-to-point wiring. Most important for the thesis at hand is the CAN's message structure. To provide a better understanding, additional background information is given, for example the electrical properties. Developed in 1983, the German company Robert Bosch GmbH is still extending the CAN specification in order to improve the quality of automobiles thereby making them more reliable, safe and fuel efficient [42].

Bus structure

The standard defines four layers (application, physical, transfer and object layer) which correspond to the abstraction layers in the open systems interconnection (OSI) model [43]. Since CAN is an event-triggered controller network serial field bus communication, it exhibits the associated topology (cf. figure 2.2), so that two wires connect all participants [44]. Principally, all of them are directly connected to each other as the information is transferred to a so-called broadcast communication, which means that the transmitted information is received by all ECUs also known as nodes [45]. This configuration guarantees a simple extension or reduction of an existing embedded system, hence, a new member can just be added to the bus lines, as long as the bus load does not tend to cross the limit of other nodes.

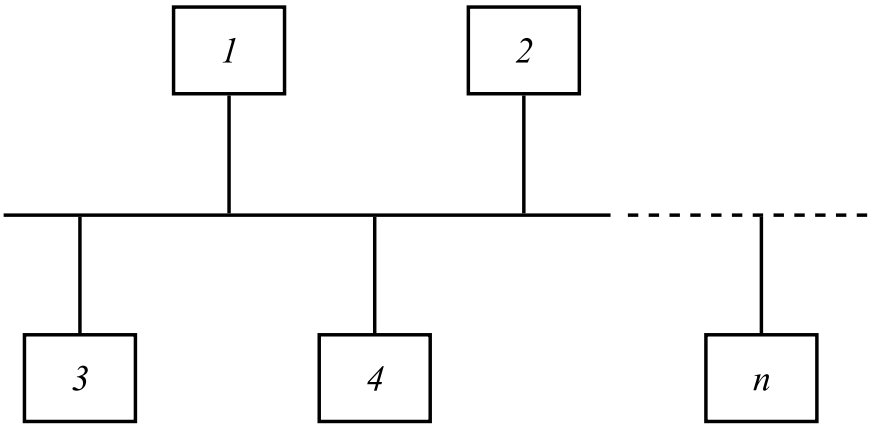


Figure 2.2: Serial field bus topology.

CAN architecture in automobiles

Typically, an automobile network is formed by different on-board buses around specific domains, namely connecting cabin control functions (e.g. window lifters, heating and ventilation system) and the powertrain [46]. The latter provides the interlinked functioning

of safety-critical components, such as engine management, anti-lock braking system (ABS) and electronic stability control (ESC). Nowadays, the requirements of multimedia, scalable, safety and security-related functions within automobiles is tending upwards in response to increased demands. This field is dominated by other bus systems like the media oriented systems transport (MOST) [47], local interconnect network (LIN) [48] and FlexRay [49], to improve the quality of automobiles thereby making them more reliable, safe and fuel efficient.

Message structure

A CAN message is packaged in a CAN bus specific format. This packaging is referred to as a frame and consists the message identifier (ID) indicating who is intended to read information and data (cf. table 2.2). Depending on the frame format (base or extended), the identifier can be represented by 11 or 29 bits. The underlying message structure is equivalent to both the standard and extended CAN version. As the data field length can be varied, it is necessary to transmit a variable that encodes this information. In addition, some standard indicators are used, such as control field, cycling redundancy check (CRC) and acknowledge field (ACK).

Start Frame	**ID**	**Control**	**Data**	**CRC**	**ACK**	**End Frame**
1 bit	11 or 29 bits	6 bits	up to 64 bits	16 bits	2 bits	7 bits

Table 2.2: CAN bus data message structure.

When it comes to a CAN system, no master is controlling individual CAN nodes that have permission to write and read data on the bus [50]. When an ECU is ready to broadcast data, on the first place, the bus is checked on the condition of being busy, only then a message is written onto the bus network. The frames that are sent over the bus typically do not contain information of either the sending ECU or any of the intended receiving nodes, on the contrary, a uniquely provided ID labels the frame throughout the CAN. All nodes associated with the network receive the frame, and depending on the frame's identifier, each ECU on the network decides whether to reject or accept the received data. In the case where different nodes are trying to send a message onto the CAN bus simultaneously, the node with the uppermost priority is automatically granted bus access [51]. Before trying to broadcast again on the CAN, nodes with low priority must wait until the bus becomes available. In this way, deterministic communication among several nodes is ensured over the CAN bus.

Electrical and mechanical properties

An electrically implemented arbiter guarantees that messages with the highest priority are transmitted earlier than those with less priority. Due to the broadcast nature of the CAN bus, all ECUs receive a message and decide whether the content is of interest [52]. The messages uses a scheme of bit-wise arbitration to control access to the bus, and each message is tagged with a priority. The lower the associated CAN identifier, the higher its priority. Low-priority messages are overwritten and send cyclically until they are acknowledged by the recipients. Data is written onto the bus which is instantiated by two or three cables (the signals are often represented on differentially transmitting cables and an optional ground wire). Rules regarding voltages and mechanical properties of connectors which have to be used do not exist within the primal norm but some quasi-standards evolved over time. Regarding some questions the norm document is ambiguous. Neither the logical dominance nor the related physical values are defined. Several implementations are used by engineers. Zero-dominant and recessive solutions can be found, mainly using 3.5 or 5 volts defining the necessary voltage level [53]. When nothing is transmitted, the recessive voltage level can be measured on the bus wire. These design decisions have consequences regarding the abilities of this type of bus system: Data rates from 125 kbit/s to 1 Mbit/s are possible when using a CAN bus cable that is not longer than 40 meters. However, larger payloads improve the protocol efficiency and lead to a higher throughput, targeting an average data rate of 2.5 Mbit/s [54].

2.3 Automata

This section briefly introduces fundamentals of automata as it deals with designing abstract self-propelled computing devices that follow a predetermined sequence of operations automatically and needed for considerations in later chapters. The most important distinction has to be made between the two types of automata which are named after their inventors: Mealy and Moore [55]. The differences concern the interpretation of transitions and states. When talking about a Mealy machine, a transition represents the input or output of a symbol while a state implies the history of states or symbols, respectively. A Moore automaton defines them contrarily: A state stands for a symbol's absorption/emission while a transition only depends on the past events. Figure 2.3 shows an example machine, specifically an input-output state machine, meaning that every transition is associated with two symbols, where for each possible input every given state has one, more than one or no transition.

If an input matches a transition for the currently present state, the machine changes to the state that transition points to. Such a machine works as an acceptor and emitter simultaneously: A validated input sequence is responded with a corresponding sequence of output events. In order to model collaboration between components of groupware systems and their interconnections, in figure 2.3 the automaton depicts all relevant properties of such a construction. These formalized properties of automata are highlighted in the following section.

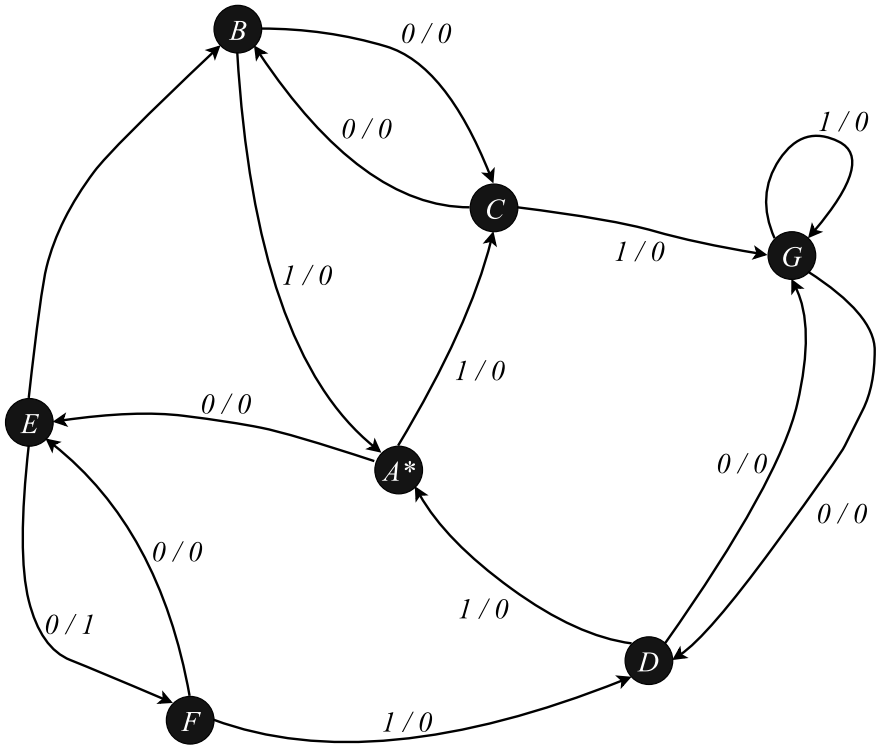


Figure 2.3: Example input-output state machine automaton.

2.3.1 Definitions

An automaton is a tuple [56] exhibiting the form:

$$\mathbf{a} = (\mathcal{Z}, \mathcal{A}, d, z_0(, \mathcal{Z}_\mathrm{F})) \tag{2.1}$$

Mathematical terms are stated subsequently following the rules presented in section 2.1.

- **Letter** $l \in$ **Alphabet** $\mathcal{A}$ – A letter is an element of the alphabet. Every symbol equals one letter.
- **Symbol** $s \in \mathcal{A}$ – A symbol is emitted, absorbed or both when a transition or state is activated, respectively (cf. [55]).
- **State** $z_n \in \mathcal{Z}$ – The black nodes depicted in figure 2.3 are called states. z_0 is the starting state, while $\mathcal{Z}_\mathrm{F}$ contains all final states in case of a finite state machine.
- **Transition** $t_{ij} \in \mathbf{T}$ – Two states are connected by a transition represented by an arrow in the picture. More common is the denotation as transition function $d : \mathcal{Z} \times \mathcal{A} \longrightarrow \mathcal{Z}$.
- **Word w** $\in \mathcal{L}$ – A word is a sorted tuple of symbols absorbed or emitted by an automaton.
- **Language** $\mathcal{L}$ – The set containing all correct words (related to the automaton) is called its language. It is a regular language *iff*: $|\mathcal{L}| \in \mathbb{N}$.

Finite State Machines

There are several types of automata. An utterly important one is the finite state machine (FSM), whose special property is the existence of finite states which represent ending points of words and reacts to occurring events [57]. The nature of the reaction depends on the currently active state and the event. Possible responses are performing actions that change variables or the system to a different state. Thus, a lamp, for example, can be situated in the state of ON or OFF, in this case, a switch would trigger events enabled and disabled. Usually, FSMs are represented with the aid of models to define the operation in a clear and comprehensible manner. A standard FSM has one starting state. Another important aspect is the property whether only one symbol can trigger a change, or if probabilities define which symbol is used. According to these characteristics, they are called deterministic finite automaton (DFA) and non-deterministic finite automaton (NFA), respectively. Most systems can be realized with the help of FSMs, however, it is certainly appropriate whenever the complexity of their functionality exceeds a certain level, to rely on an advanced model. Specifically, the adoption of nested states often facilitates the modeling of complex systems.

2.3.2 Properties

Equivalence

A minimal representation exists for every system. Figure 2.4 shows the reduced automaton representing the same system as the state machine in figure 2.3. The property which has to be equivalent is the set of valid words. The example can be retraced when state A* is understood as the given starting point. Since the depicted automata are based of both inputs and outputs, an accepted word has to result in the same outcome, i.e. a chain of ones and zeros. Figure 2.4 shows that a smaller (5 states and 10 transitions) and a bigger automaton (q.v. 2.3) (7 states and 14 transitions) can represent an equivalent (identical) system behavior.

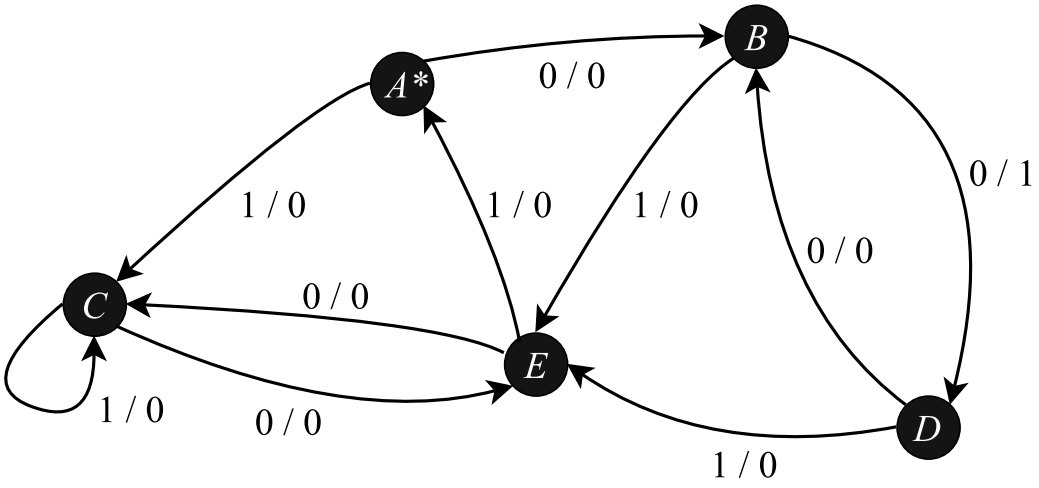


Figure 2.4: Reduced example input-output state machine automaton.

Definition 2.3.1. Homomorphism: This term implies the relationship between two automata, when both accept the same language but do not show equal properties concerning the structure, i.e. states and transitions. The mapping described has to be valid in one direction.

Definition 2.3.2. Isomorphism: When the mapping mentioned above is invertible, two automata with the same language can be considered isomorph. Isomorphism exists between two machines, if they differ in the their state's name and otherwise exhibit the same behavior.

Graph properties

The theory of graphs provides several useful attributes for the description of automata [58]. All of them can be used for complete graphs as well as for subgraphs, thus for only some of the objects and links. Every automaton is a directed graph, for the reason that a transition works only in one direction. It can be symmetric but in most cases it is considered as asymmetric, otherwise there would have to be either two or no vertices between two states. Concerning the structure of a graph, cyclic graphs (also: circular or cycle graphs) can be defined by the fact that all states are periodic, because all of them lie within a circle structure. A graph is called recurrent, when all objects are reachable several times in a finite amount of time, while states which can be reached only once are called transient. If a subgraph is cyclic and is defined by a subset of states which cannot be left, it is called ergodic. When the subset has only one member, it is denoted absorbing. A good possibility to evaluate the inner connectivity are the following attributes: A directed graph is weakly connected, if all objects can be reached from every other state, without respecting the direction of the transitions. If this is the case when the directions are respected, the graph is strongly connected. A graph which is strongly connected cannot be reduced and is therefore called irreducible.

Discrete event-based systems

Modeling of a system in discrete event-based manner is the application of automata theory driven by electrical engineering [59] and automotive systems [60]. Other common tools are Petri nets [61] and Markov models [62–64]. The most important characteristic are time-dependent states that are modeled with real-time requirements. A possible application field is usually the representation of dynamic systems that are modeled by finite state automata. Petri nets, named after Carl Adam Petri, are an extension of standard automata, providing possibilities to represent parallel and/or dependent processes with less objects. Another common name is place or transition net which imply the basic principle of such nets: Three types of objects are necessary to build a Petri net, namely transitions, directed arcs and places. The arcs connect a transition and an output or input place depending on its direction. Every place can be marked by one or more tokens. Transitions fire as soon as all of their input places are marked by at least one token and transfer them to the output places. Since a transition stops the ongoing of the process until all necessary places are marked, it is most certainly that parallelism and dependencies can be modeled in a straightforward manner.

Hierarchies

Automata which represent complex systems can have a very large extent. It is possible to describe systems of any complexity, but with increasing size it can be reasonable to introduce hierarchies. Two ways of abstraction are conceivable: The different levels can map actually existing system levels, so that the machine on the uppermost level represents the overall system. The second possibility is to use several automata to distinguish between different time domains. In the second case mentioned before, one machine is running for a defined period of time until it is replaced by another machine representing the subsequent period of time. Both possibilities explained use a number of smaller automata to describe a larger system. As a consequence, a connection between the smaller machines becomes necessary. This can be solved by introducing a normal transition or by using a completely different technique.

Canonicity

When automata are intended to represent a certain structural behavior, some further criteria have to be defined. Otherwise canonical automata will be the straightforward solution. Without stating constraints concerning quantity of states and transitions or complexity, the easiest suggestion would be a maximal or a minimal canonical automaton. The first type exhibits the following properties: One starting point leads to n ($n = |\mathcal{L}|$) sub-chains, each is as long as the word that it stands for (cf. figure 2.5).

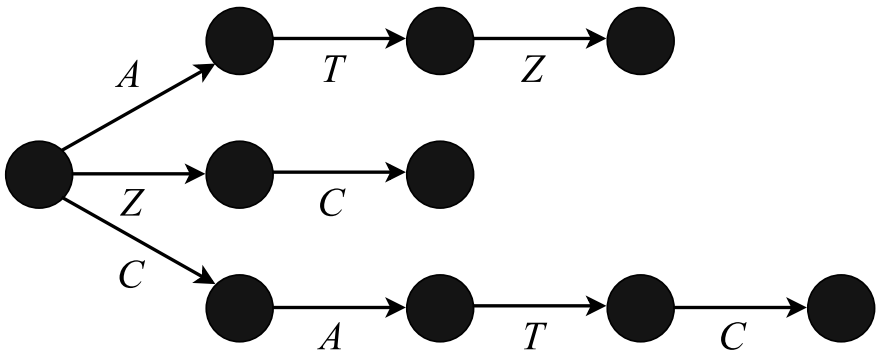


Figure 2.5: Maximal canonical automaton with ten states and nine transitions.

A minimal canonical automaton is created even with less effort. It consists of one state node with m ($m = |\mathcal{A}|$) transitions pointing at it (cf. automaton with one state in figure 2.6).

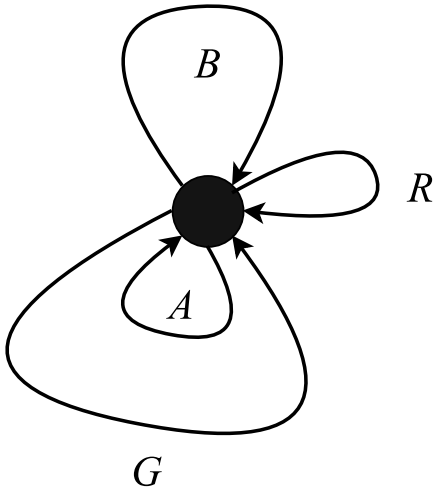


Figure 2.6: Minimal canonical automaton with one state and four transitions.

2.4 Statistics

Statistics can be considered as the fundamental basis of this thesis. The used standard formulae and methods [65] are listed in the following itemization in order to introduce the topic.

Standard metrics

The simplest but probably most important metrics in statistics when analyzing a sample set $\mathcal{X} = \{x_1, ..., x_n\}$ are:

- **Arithmetic mean** – The arithmetic mean $\overline{x}$ is commonly referred to as an average, obtained by adding quantities together and dividing the sum by the quantities' value.

$$\overline{x} = \sum_{i=0}^{n} (\frac{x_i}{n}) \tag{2.2}$$

  where $\overline{x}$ is the arithmetic mean, x_i represents a group of values and n is a total of numbers in a data set. The peak of a Gaussian distribution corresponds to this point.

- **Variance** – The variance represents the width of the distribution function at the base of the graph (in contrast to kurtosis) indicating how widely the existing values are dispersed around the mean:

$$s_{n-1}^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2 \qquad \text{(unbiased)} \tag{2.3}$$

$$s_n^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2 \qquad \text{(biased)} \tag{2.4}$$

  where s^2 denotes the sample variance, $\overline{x}$ is the sample mean, x_i represents the ensemble of all possible samples and n is a total of numbers in a data set.

- **Standard deviation** – The standard deviation s is the square root of the variance s^2:

$$s = \sqrt{s^2} \tag{2.5}$$

- **Coefficient of variation** – The coefficient of variation c assesses the size of the standard deviation s in relation to the size of the arithmetic mean $\overline{x}$ of the population:

$$c = \frac{s}{\overline{x}} \tag{2.6}$$

- **Linear regression** – The main goal is to create a model/function h_θ that is trying to map the input x, namely the independent variable to an output data $\hat{y}$, called the dependent variable. We assume that the numeric output is the sum of a deterministic hypothesis function of given model parameters θ_0 and θ_1:

$$\hat{y} = h_\theta(x) = \theta_0 + \theta_1 x \tag{2.7}$$

- **Cost function** – When referring to a minimization problem, the cost function $J(\theta_0, \theta_1)$ measures the accuracy of the hypothesis function by taking an average of given inputs x compared to the actual output y. The overall objective is to minimize the parameter values θ_0 and θ_1 so that our mean $\overline{x}$ of the squared difference of x and y is rather small. As a computational convenience, the mean of the total number of training examples m is halved $\left(\frac{1}{2m}\right)$ so that the derivative of the used square function will cancel out the $\frac{1}{2}$ term.

This function is also known as the squared error function or mean squared error function:

$$J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} (\hat{y}_i - y_i)^2 = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i)^2 \tag{2.8}$$

- **Gradient descent** – Used for automatically minimizing the cost function and improve its parameters θ_0 and θ_1:

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1) \tag{2.9}$$

where α represents the learning rate number and controls how aggressively gradient descent is and $\frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ denotes a derivative term which updates the parameter values θ_0 and θ_1 simultaneously.

When applied to a regression problem, gradient descent can be derived as follows:

repeat until convergence: {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) \tag{2.10}$$

$$\theta_1 := \theta_1 - \alpha \frac{1}{m} \sum_{i=1}^{m} ((h_\theta(x_i) - y_i) x_i) \tag{2.11}$$

}

where θ_0 is a constant that changes simultaneously with θ_1, m is the total size of the training examples and x_i, y_i are specific values of a given training set.

- **Central moments** – The first moment equals the mean, the second central moment is the variance. While the third standardized central moment is the skewness, the fourth standardized central moment denotes the kurtosis.

- **Kurtosis** – The kurtosis measures the dispersion of the whole distribution around the mean. In figure 2.7 the kurtosis is low in the upper diagram on the left, while it is high in the diagram on the right. The higher the kurtosis, the larger the part of the variance that is a result of infrequent extreme deviations, i.e. there are more outliers in the data.

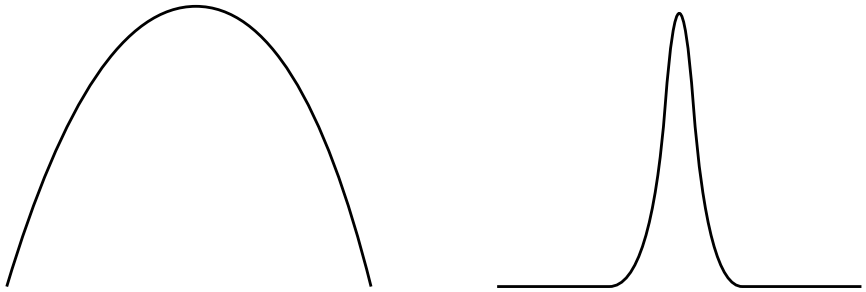


Figure 2.7: Low and high kurtosis on distorted normal distributions.

- **Skewness** – This parameter indicates whether the variables are distributed evenly and symmetrically around the sample mean. In the left diagram on the top in figure 2.8 the skewness is positive, in the diagram below it is negative.

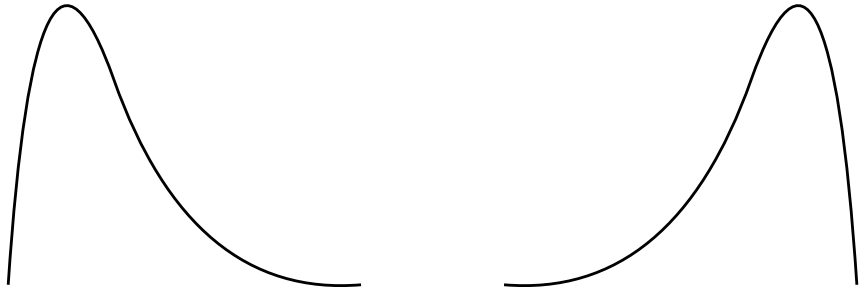


Figure 2.8: Positive and negative skewness on distorted normal distributions.

Principle Component Analysis

The principal component analysis (PCA) [66] provides a possibility of minimizing the property dimensions of a data tuple, so that the membership to a certain class still can be defined with certainty. In the end, it is a geometrical technique. The coordinate system can be transformed so that one or more axes are removed, if an equivalent clustering is possible without the axis or axes in question. A more detailed introduction to PCA is given in section 3.2.3.

Part II

Theory

Chapter 3

Outliers and anomaly detection

This chapter emphasis the properties of data recordings from observing vehicular bus messages, introduces the term anomaly and relates it to the terms error, fault, and failure. It further surveys different techniques commonly used in the area of anomaly detection and diagnosis.

Anomaly detection, as concerned in this thesis, addresses the problem of recognizing and exposing abnormal behavior when observing the data exchange of in-vehicle network bus messages in a modern automobile.

An abnormal behavior of a system is referred to as anomaly [67]. Alternative names similar to the term anomaly used in research are outlier, novelty [68] and discordant observations [69]. Anomaly detection basically implies noticing what does not normally happen. For example, in a data set of credit card transactions, it may indicate fraud [70]; in health care, an outlier may express a significant deviation from a patient's normal behavior [71]; in image processing, anomalies may point to abnormal patterns or textures, for example, handwritten digit recognition [72]; in network security an anomaly may reveal intrusion attempts in a data set of network traffic [73]; in automotive security due to observed fraudulent injected bus messages or errors during test drives to detect anomalies from in-vehicle networks [74], [75].

3.1 Taxonomy

Essential terms which frequently occur in the area of anomaly and fault detection have to be clarified. The description of anomalies from [76] is adapted in order to describe outliers from vehicular network CAN bus messages.

Definition 3.1.1. Anomaly: In the following chapters this term is understood as behavior deviating from regular test cases without malicious CAN data packages.

Definition 3.1.2. Error: The difference between the true (i.e. real) value of a variable and the value measured directly or indirectly, is known as error.

Definition 3.1.3. Fault: A fault is the existence of a gap between the specified and the actual behavior observed.

Definition 3.1.4. Failure: The status of a system when it is not working any more, without further detailed definition of cause or consequences, is called failure.

Definition 3.1.5. Detection: Recognizing whether a specified event (f.e. a failure) is taking place in a certain moment is referred to as a detection.

Definition 3.1.6. Diagnosis: Detection including an interpretation of the recognized event, i.e. determination of location and type, equals diagnosis.

Definition 3.1.7. False positive: When a judgment rating something as positive is wrong, this is a false positive.

Definition 3.1.8. False negative: When a judgment rating something as negative is wrong, this is specified as a false negative.

Definition 3.1.9. False reject: This term denotes the fact that an attribute is rejected even though it should be accepted. In the following, this is the case when an anomaly is detected in an actually correct set of symbols.

Definition 3.1.10. False accept: This term denotes the fact that an attribute (e.g. set of symbols) is accepted even though it should be rejected.

Definition 3.1.11. Malfunction: A fault which is present only temporarily, possibly emerging periodically, is termed a malfunction.

Definition 3.1.12. Symptom: The deviation of an observable variable from expected behavior is known as a symptom. The effect does not have to have any logical connection to the source, thus it can be very challenging to trace back the explicit malfunction to its cause.

3.2 Concepts

Anomalous states in vehicular networks can be detected in two different ways. On the one hand, anomalies can be detected from the raw data stream or based on features extracted from data itself. For the purpose of reducing the dimensionality, raw data can be preprocessed [77] on the other hand data can be transformed to an alternative representation [78]. Since the objective of this thesis at hand is the detection of possible vehicular network intrusions by recognizing anomalies, common anomaly detection methods are described in this section. While methods which already found their way into applications are presented, also approaches which are in the focus of current research are discussed. In every subsection, the difficulties and restrictions of the methods are illustrated (these considerations are completed in the subsequent chapter). A final remark of introduced concepts is given in paragraph 3.4.

3.2.1 Conventional

Methods which make use of knowledge regarding the relationship between values enabled by human reasoning, can be described as conventional detection methods. Mainly variables defined within a closed system are observed to derive a judgment regarding the system's state.

Logical reasoning and thresholds

An utterly common approach in the area of detecting anomalies is the observation of all variables involved [79]. Thresholds have to be defined a priori, so that an anomaly is evident when a value exceeds or falls below a limit. A considerable disadvantage of this method is the fuzziness of the thresholds and the resulting fuzziness of the detection. One possibility to improve the technique is to add a logical reasoning, which has to decide if a combination of values is reasonable or not. By doing so, system states can be defined depending on different parameters measured. Redundancy in measuring can guarantee that the estimated state is true; for example, the statement that a car is driving can be verified by an internal

velocity merit but also with gyroscopes integrated in other components. This can help to preclude harmful behavior of the system's software, e.g. violating previously fixed conditions.

Observing the development of values

In [80] a method is developed which allows to recognize and understand the development and trend of qualitative (observable) values. The term qualitative values, that is introduced, already implies the basic idea: The abstraction of information contained in the variables. The most important objective is to reduce redundant data. The arising qualitative model is then used for a real-world application, namely the detection of emission-related failures in a passenger vehicle. In this scenario, a model-based reasoning method uses the qualitative values for the judgments. In conclusion, this approach combines the later mentioned methods of formalized software verification (cf. [81]) and the observance of values described above.

Fault screening

Abreu et al. [82] present a method of verifying a symbol stream by using fault screeners. A Bloom filter, a bitmask and a range screener are discussed and utilized for spectrum-based fault localization. The main idea of these methodical suggestions is the use of generic invariants while the performance overhead is kept as low as possible to ensure computational efficiency for embedded systems. The three techniques work as follows:

- **Bitmask screener** – This method observes a string of bitwise representations of system variables. Due to the bit level operations it is a considerably promising tool.
- **Range screener** – In analogy to the conventional fault detection mentioned above, variables are observed regarding the violation of thresholds.
- **Bloom filter** – In this case values of variables are bit masked, furthermore, mapped into an allocated memory space by two hash functions. If an address was not written during the initialization phase, the value is invalid. Due to the used hash functions some of the unique addresses can be mapped for different values. This results in memory optimization, but it also involves probabilistic uncertainties of the Bloom filter's judgments.

The evaluation of the above mentioned methods leads to the result that two of them can be used for anomaly detection, but show different performance depending on the scenario. The bit mask screener is identified as the least favorable working concept, while the range screener works more promising than the filter except for cases with a huge amount of data.

3.2.2 Operational

The concepts mentioned in the following have been developed in the environment of software engineering to provide tools to test software after the development process is completed. Usually, a detailed software documentation is necessary in order to enable their usage.

Software verification

Formalized methods known from software engineering provide ideas and concepts which seem to be helpful in solving the problem of this thesis (cf. chapter 5). The fundamental basis of the presented techniques is a model representing the system. There are several possible methods to derive such a model. In [83] and [84] two of them are presented. The main reason why the concepts cannot be applied as they are, is the fact that information from and about the development process is needed [85], moreover, the representation of not yet observed states, e.g. future states for which predictors are available, is necessary.

Conformance testing

Conformance testing is a broad field with different methods that are commonly used [86]. In contrast to software verification, the system is tested using a standard where the aim is to check the system to a given specification. The question which has to be answered is whether the system meets this norm or not. Nevertheless, it cannot be applied to the problem discussed in the thesis at hand due to the same reasons as described in the paragraph above.

3.2.3 AI-related

Different strategies known from artificial intelligence (AI) already found their way into the field of anomaly-based intrusion detection. Approaches that can be used for raw CAN bus data have been proposed in the literature that are presented and discussed in the following.

Artificial neural networks

An ANN is a sum of formulas weighting different properties of an object which has to be classified [87]. For supervised learning tasks, these weightings can be adapted, i.e. learned so that objects are labeled when the application is executed. In anomaly detection this labeling is either a reject or an accept. One problem that arises is the synchronization process that is necessary to get the ANN running. A more detailed view on ANN is given in chapter 4. In [88], a concept is presented using an ANN that is trained with parameters based on features that are derived from a CAN trace. More precisely, an artificial network for given CAN data provides the likelihood of classifying normal and malicious data, and, thus the system can recognize any potential attack such as network intruders or deliberately injected packets.

Data mining

The term data mining is used to describe the systematic application of statistical methods to large data sets, also known as "big data", with the aim of identifying new interconnections and trends to gain benefits from data [89]. Such assets can not be processed manually, so one needs computerized methods because of their size. The common approach is to classify systems by directly observing properties or invariants. This means that values are represented in a multidimensional feature space and thereafter data mining techniques are exerted. Furthermore, visualization can be used for various data mining tasks such as cluster detection, classification or pattern discovery, and hence can be used for anomaly detection in IDS [90].

Principal component analysis

Introduced in chapter 2, PCA offers a possibility to reduce the amount of attributes required to classify an object. However, one drawback arises when trying to apply this technique to the problem described in chapter 5. In contrast to the data which can be directly used for the issues at hand, the amount of information is immense. Since every system can be

described in an abstracted way, it can be represented by data points in several dimensions. If there were a lot of dimensions spanning this configuration space, it would be possible to use a PCA to determine which dimensions define the system sufficiently. Since only timestamps and associated data are accessible values, using a PCA would not offer a substantial benefit. However, if a lot of positive traces are available, an arbitrary modeling technique can be used to map the system. When a metric is employed to rate the resulting model, it can be used as an additional dimension. As a result, traces that are necessary for a sufficient learning can be evaluated. As only a few recorded CAN traces are available, this concept is rejected.

Instance-based techniques

Instance-based techniques such as k-nearest neighbor (k-NN) are used to classify unseen data to one of the classes normal or abnormal [91]. As with many non-parametric algorithms, instances are represented as points in a multidimensional feature space. An instance is defined on the basis of the data's attributes. Each attribute represents an axis in the multidimensional space, where the number of occurrence of the attributes is merged into a point vector. The decision is made in the k-NN method, as the name already clearly suggests, to classify the new reference to the closest data point. Classification is based on the distances to instances in a knowledge base. Either the entire time series or feature vectors can be stored in the k-NN knowledge base. The calculated distance can then be used as an anomaly rating.

Rules or decision trees

A decision tree is a representational form for a classification rule, by means of where objects can be divided into classes. Used for classification and regression, decision trees can be used as a non-parametric learning technique. A tree is designed so that an attribute in each node is queried, and a decision is made, until a leaf is reached. A leaf represents a node at which no further branching is performed. This is where the classification can be read, therefore, rules and decision trees are also called classification trees. The foundation for building a decision tree is a training data set whose class membership is known. In general, the learning examples are characterized by a fixed number of attributes, each having a finite number of forms and a classification which indicates the possible decisions. In contrast to pattern recognition, decision trees deal with discrete attribute values. According to the interval logic in [92], temporal classification rules can be discovered [93] and a decision tree can be created.

Clustering

Classification techniques such as k-means algorithm can be used to find clusters of normal data [94] and detect clusters between sequenced events and subsequences respectively. K-means provides a classification in groups, which have according to a global level, optimal homogeneity among all groups, having a predetermined number of clusters. The discovery of hidden relations between event sequences involves following steps, namely: the representation of the data in an applicable form, the interpretation of analogies between sequences and subsequences, and the utilization of derived models to substantial problems in data mining.

Hidden Markov models

A hidden Markov model is a representative statistical model that can be used to model data which is sequential in nature. In a hidden Markov model (Markov process), a finite set of states can be trained to determine whether the order of subsequences is normal or abnormal [95]. It consists of hidden states X, possible observations y, a denoting state transition probabilities and output probabilities, characterized by b. Each of these states at any time emits a randomly selected visual symbol or so called visible state. To an external observer, only the outcome, not the states that is visible, hence, states are hidden to the outside. Hidden Markov model strategies may also be used for detecting possible intrusions by predicting the hidden state (attack) from relevant observations such as changes in system parameters, fault frequency, etc. In [96] an approach is proposed that uses a multivariate Gaussian model. The proposed system correlates the usage and activity profile observations and state transitions in order to predict the most probable intrusion state sequence.

Autoregressive-moving average (ARMA) models

ARMA models [97] can be used to fit a model to either the entire time series data or subsequences. When dealing with non-static series, taking the transformation to stationarize the series, anomaly detection can then be done based on distances between the model parameters or on the probability that a sequence is created by a certain model. The main properties of an autoregressive and a moving-average model is the correlation between time series objects at a given time point. In [98] it was shown to work, but is not viewed as a promising approach since the order and the coefficients of the models have to be determined.

3.3 System modeling

An important issue in anomaly detection is the system modeling that is used as a basis for verification. A convenient but also time-consuming possibility to model a system manually. This includes several steps in an effortful process. All possible states and possible transitions between them have to be identified and all relevant input has to be defined. Information is extracted from the system specification and documentation as well as by observing the system's behavior. A system modeling of this kind requires a lot of knowledge regarding actual and intended behavior. An expert involved in development and testing is mandatory. Model-based diagnosis (MBD) summarizes the methods mentioned in the previous subsection, since they all base on a model representing the system being observed. Often automata (q.v. 2.3), Petri nets (q.v. 2.3.2) or other abstraction tools are used to represent the systems.

3.4 Summary

This chapter discussed properties of anomaly detection, introduced related terms and approaches to automata learning. Summarizing the descriptions above with regard to the actual problem, which has to be solved by the thesis at hand, the following can be assessed. Systems used for the purpose of detecting anomalies that are independent of architectural details do exist but do not apply entirely to the problem presented in the following chapter. Conventional techniques need a lot of information regarding the system. Even if every piece of information is available, most of the modeling has to be done by hand, if it is not accessible in a formalized manner. Other deficiencies can be seen concerning the use of hidden states, features and structures. The substantial problem that concerns the methods entirely is the fact that specifications or hand-modeled system representations do not contain all system states which exist in reality, because further information regarding the system's model were not accessible in the development process and would exceed the limits of this thesis at hand. A method which manages the learning of an overall distributed system from malicious data packets, represented by future behavior depending on past messages, could be found. For predicting and classifying packet data, ANNs had been investigated by [99–101]. Since a user-driven approach on modeling a large data set does not suite well to the aim and actual problem (cf. section 1.1), an approach capable of learning from sample data is necessary. Therefore, machine learning and classification theory are discussed in the following chapter.

Chapter 4

Intrusion detection as a classification problem

This chapter discusses intrusion detection using classification techniques. The topic intrusion detection is surveyed, fundamentals of machine learning are given, and artificial neural networks are introduced.

Research works considering safety problems in vehicular network communication is a long-standing objective in the security field [102]. IDS gain much attention due to the ease of detecting attacks caused by intruders [103].

In [104], an IDS approach is proposed where numerous applicable attacks, predefined in a database, are applied. According to [105], a method, based on specifications, comparing the specification system's current behavior to the predefined patterns, is developed. As stated in [106], various sensors, designed for attack scenarios, are used for detecting network intrusions. In [107] as well as [108], protocols that are secured under the terms of commonly used specifications are proposed. In order to deal with the emerging problem, statistical-based IDS techniques are considered to a large extend for communication networks. In [109] statistical features are captured and applied to detect cyber attacks [110].

4.1 Intrusion Detection

The goal of intrusion detection is to filter out all events, taking place in the monitored area, those that indicate attacks, attempted misuse or security violations. An IDS can be defined as a file of tools, methods and sources helping reveal, record or announce the attempts on network intrusions and attacks [111]. IDS operate on the network layer and usually they are passive systems which are not aimed at attack prevention. The primary purpose of IDS developing is to reveal the attack, not to execute appropriate measures to avoid it. Systems trying to avoid attacks are called intrusion prevention systems (IPS). With regard to the development, some of the functions have grown up to prevention measures against attacks, and then several IDS become active. They were renamed as intrusion detection and prevention systems (IDPS) [112]. Any use of prevention functions should be indicated in IDS records. Patterns of abnormal behavior are many; however, in general, they include unneeded, harmful or illegal activities occurring within the system. We distinguish two main methods of intrusions detection, i.e. misuse detection and anomaly detection [113]. According to the principle of activity, intrusions detection can be divided into three groups: signature-based IDS, anomaly-based IDS and hybrid systems (cf. [103]).

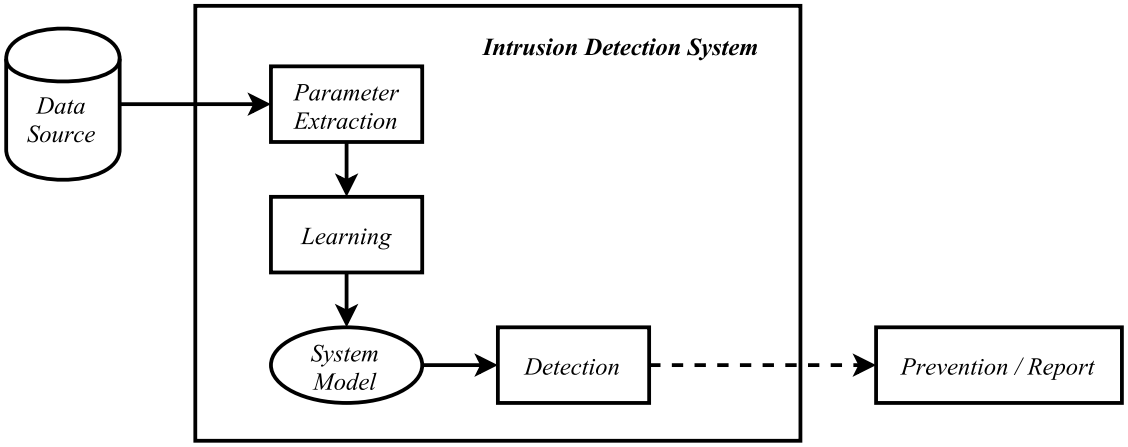


Figure 4.1: Simplified schematic of an IDS using statistical analysis.

Anomaly-based IDS take a different approach to intrusion detection. By analyzing/learning network traffic and processing the information with numerous statistical algorithms, anomaly-based IDS look for anomalies in the normal network traffic patterns, illustrated in figure 4.1.

4.2 Machine learning

Algorithms and ideas from the field of machine learning (cf. [87]) are closely related to the field of statistics and help refining algorithms as well as understanding dependencies in data. The commonly applied steps in a machine learning based system are illustrated in figure 4.2.

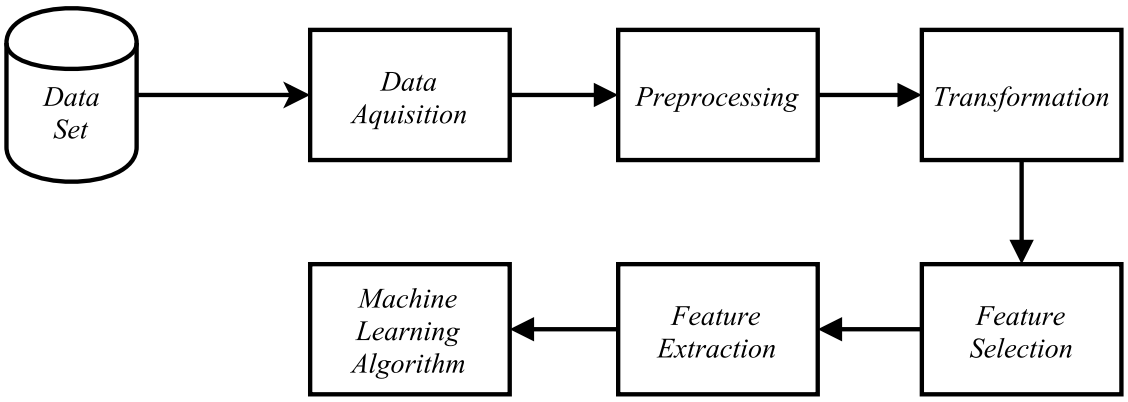


Figure 4.2: Common steps in a machine learning-based system.

The first action is the data acquisition step, which comprises obtaining and selection of data sets. Optionally, the input data can be preprocessed, for example by re-sampling or removal of invalid data. Following that, a further optional step is the transformation of the data set to an alternative representation that, for example, allows for better distinction of classes or for more efficient processing. Examples are the mapping of the floating point values of a time series to a limited number of digits or symbols, e.g. using HOT SAX (cf. [69]), or the transformation to frequency domain using Fourier or wavelet transformation. Subsequently, a vital step is the extraction of features, a machine learning algorithm can work on. As a final step, the features relevant for the given machine learning task are selected. Today, this scientific field is emerging and conquering more and more areas. It combines different disciplines, furthermore, it becomes increasingly popular due to the vast possibilities in applications such as computer vision [114], natural language processing [115], robotics [116], video games [117] and search engines [118]. The long term goal is to imitate learning behavior of biologic systems like the human species. To this day, software is merely inter- and extrapolating data points that represent change over time. Very important definitions taken from [119], which are confused frequently, are given in the following listing and paragraphs:

- **Live/preprocessed learning** – This modus defines whether the learning takes place when the system is running in its application environment or under laboratory settings.
- **Online/offline learning** – This term is used differently depending on author and field of research. In this thesis, it has to be understood in connection with the learning process: offline means that the learner is provided with observation sets which are categorized. Online signifies that the learner is continuously supplied with sets and shortly after it with associated labels (cf. reinforcement learning).
- **Active/passive learning** – This distinction is made regarding the question whether a learner can influence data source. A good example is the simulation of an input to measure the related output for example a black box principle.

Overfitting

The phenomenon of overfitting occurs if a learned model fits the data used for the learning progress too well. This means that for example a mathematical function, which describes the data very well or even too well, cannot explain other related data. A lot of effort has to be put into the decision which level of detail has to be represented by the model at maximum.

Pattern recognition

Pattern recognition can be considered as the part of machine learning which concentrates on recognizing and understanding patterns in data, e.g. pictures, emails or abstracted information. For this purpose, features have to be defined which are responsible for the creation of a pattern. For example, edges and corners are often used in the field of computer vision.

Supervision

Machine learning can be separated into two main fields: The unsupervised and the supervised strategy. The latter's idea is to recognize defined properties from examples taken from a database, using them to improve the capabilities of recognizing samples. The examples are categorized by a supervisor before. Thus, regression and prediction can be classified as supervised learning problems, while clustering and pattern recognition are part of the category unsupervised learning. A highly relevant difference between those two is the possibility of verifying and evaluating the result [120], which is not possible for unsupervised problems.

4.3 Clustering

Clustering is a sub field of machine learning and can be applied in supervised learning as well as for unsupervised issues. N-dimensional data points representing n properties are separated into groups. The process of grouping, hence, the order of the iterative steps and the way of determining the distance between the points, defines the method of clustering.

Definition of distance

The most straightforward way of measuring the distance between two points is the Euclidean definition, which measures the shortest way without weighting properties. Another akin distance is the so-called city block. It defines the range by accumulating the vertical and horizontal distances, as if one would walk through streets in a rectangular street pattern. A more sophisticated approach is the Mahalanobis distance [121] which is defined as follows:

$$d(\vec{x}, \vec{y}) = \sqrt{(\vec{x} - \vec{y})^T \mathbf{\Sigma}^{-1} (\vec{x} - \vec{y})} \tag{4.1}$$

A beneficial advantage is the fact that the relationship between the different parameters is taken into account by using the covariance matrix $\mathbf{\Sigma}$ (cf. [87]). In this way, the relationship between all parameters is considered, i.e. if they are proportionally related (positive $\mathbf{\Sigma}$), inversely proportionally related (negative $\mathbf{\Sigma}$) or not connected. The covariance is given by:

$$cov(\mathcal{X}, \mathcal{Y}) = E[(\mathcal{X} - E[\mathcal{X}])(\mathcal{Y} - E[\mathcal{Y}])] \tag{4.2}$$

The variance (cf. 2.4) is a special case of the covariance, if $\mathcal{X} = \mathcal{Y}$. E is the expected value and here equals the sample mean (cf. 2.4).

Hierarchical clustering

In contrast to the conventional clustering methods, the hierarchical clustering does not need any configuration variables. There are two main classes of hierarchical clustering, namely: agglomerative clustering and devisive clustering. While the agglomerative clustering method puts smaller clusters together, the devisive clustering technique cuts one into two sections iteratively. Alternative names for these approaches are bottom-up and top-down due to the organization of the resulting clustering process. The former begins with the maximum count of clusters while the latter approach deals with only one cluster having all data points [122].

Dendrogram

The dendrogram shown in figure 4.3 is the most predominant way of visualizing a hierarchical clustering of data. A dendrogram provides an interpretable graphical representation of clustered formations where each joining combines two clusters. The advantage of this grouping is that the distances between clusters that are joined into a new group can be illustrated by vertical lines. A single linkage hierarchical clustering algorithm constructs a dendrogram from the data. Single linkage clustering can be viewed as a recursive process of pairwise merging of the closest pairs. It is this merging process that defines the dendrogram tree structure. The result is a cluster dendrogram (cf. figure 4.3) that identifies the clusters for a given linkage value. Single linkage clustering can also be viewed as the computation of the minimum spanning tree of the data. The different values labeling the x-axis represent the leaves of the clustering, i.e. the different clusters. Each of them is filled with at least one data point. Illustrated by the length and measured by the value markers of the y-axis, it is clear to see immediately which groups are separated by a extensive distance and which lie together closely. As the height represents the distance between two objects that are connected, these heights can be used to evaluate the segmentation. This means that a short vertical line connecting two leaves or the groups of several leaves, implies a short distance between them. Altough dendrograms are often considered as a technique of representing data graphically, marginal changes in the data may lead to different hierarchical clusters.

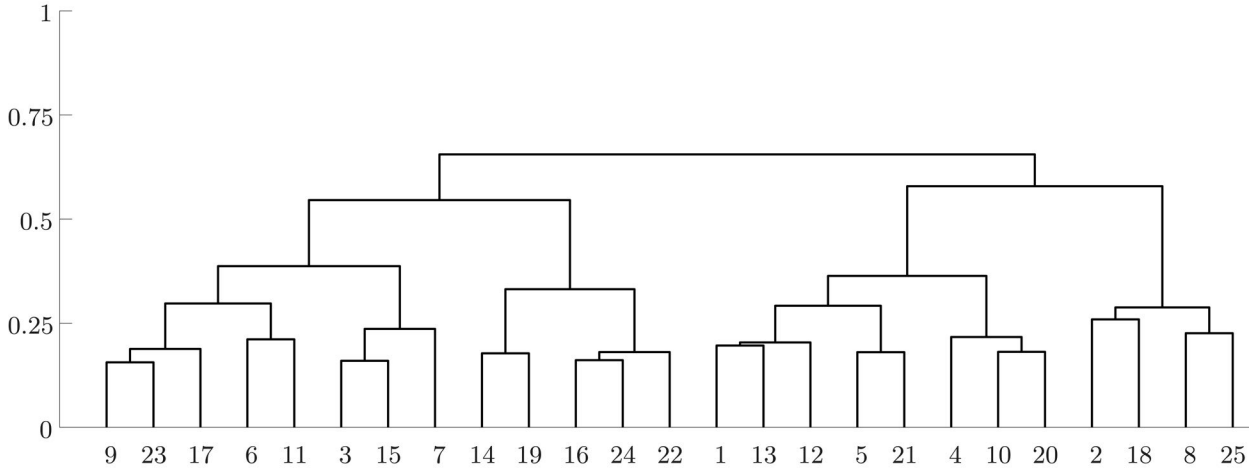


Figure 4.3: Identifying clusters in a dendrogram consisting 25 data points.

4.4 Artificial neural networks

ANNs (also known as 'connectionist models' or 'parallel distributed processing') are biologically inspired flexible, nonlinear parametric models that mimic biological neural systems [123]. The most important feature of an ANN is the ability to learn. After an ANN has learned data, it has the ability to depict skills of a biological brain as generalization, pattern recognition. An ANN is, after it is properly learned, able to divide data into classes.

4.4.1 Neuron

The structure of individual neurons is taken from the biological model. This point is well illustrated in figure 4.4. A human nerve cell gets input provided by dendrites, which is summed up and sends its output through an axon. Typically, an ANN consists of layers, with a vast number of neural units (neurons), which are represented as units in the ANN, and synapses, which are defined as connections or adaptive weights between units. The value of a weight determines the connection strength between two neurons. In order to identify the relation between the presented inputs and targets, the weights are adapted, or learned, so that the expected error of the learned model is minimized. ANNs are configurable models which allow the developer to incorporate domain knowledge into the architecture.

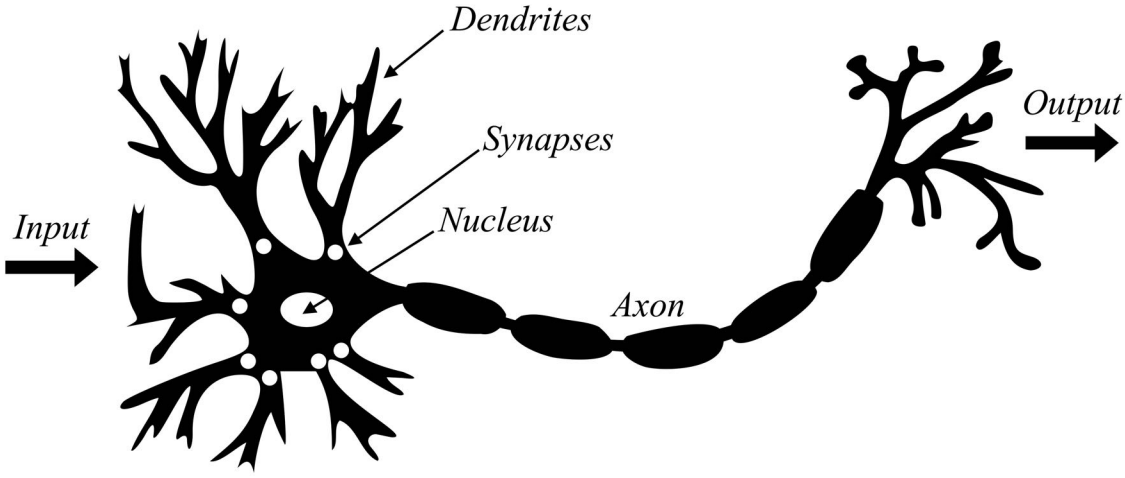


Figure 4.4: Schematic representation of a biological neuron together with its interconnections.

Definition 4.4.1. Neural network: A neural network is a tuple (M, Q, f) where M is the amount of the neurons, Q as the connection graph and f is the function of the learning rule.

ANNs have been widely applied to solve many difficult problems in different areas, including pattern recognition, signal processing and language learning. There have also been numerous applications in health care and finance, moreover, several architectures have emerged [124].

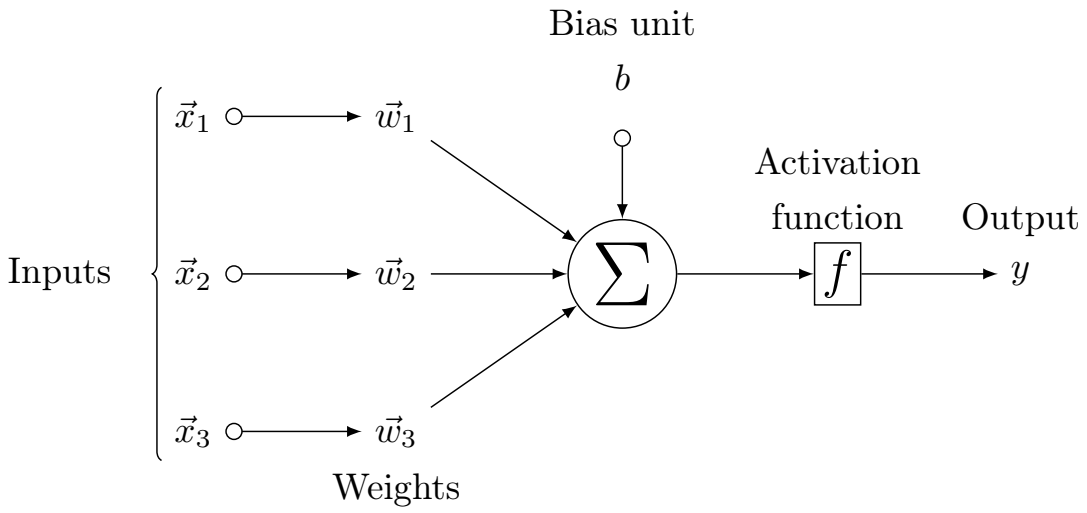


Figure 4.5: Mathematical representation of an artificial neuron model.

Figure 4.5 represents the structure of an artificial neuron model with the vector $\vec{x}$ as the output from the neurons with incoming connections considered to the actual neuron. The vector $\vec{w}$ includes the weights of the incoming connections. Σ is the sum function of resulting weighted values including a constant input of 1, called the bias unit. f denotes an activation function (e.g. figure 4.6). The calculated output y is passed on to subsequent neurons. Regarding the structure of a neuron and its associated mathematical functions there are established different perspectives and descriptions. This work is based primarily on the descriptions provided by [125]. Each neuron is associated with an activation and an output function. Using these functions is determined by how strongly the compounds are irritated. An activation function is also referred to a sum function, which adds up the incoming values.

$$\sum_{i=1}^{n} w_{ij} * o_i \tag{4.3}$$

where n is the number of neurons that have an exit towards the currently viewed neuron, w_{ij} is the weight of the i-th input neuron to the current neuron j and o_i is the output. Along with the output function, a threshold can be set from which the neuron emits an action. If the calculated result is below this threshold value, the neuron produces no output. The output is usually limited from above, so that it lies in the interval between 0 to 1. In a straightforward manner, the output function with the least effort is the binary output function, which outputs in the event of activation 1; otherwise 0. It is, however, more useful to use at least one linear output function or at best a sigmoidal [126] (S-shaped) output function. One problem arises with linear output functions, that is, linear functions are not limited, and grow the values infinitely. With regard to sigmoidal functions, the hyperbolic tangent, logistic function or sine is provided. When applying sine, only the interval between $-\frac{\pi}{2}$ and $\frac{\pi}{2}$ is used. Since the logistic function is used when using an ANN, it is discussed more detailed. In figure 4.6 the sigmoidal (logistic) function is illustrated with the following function term:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{4.4}$$

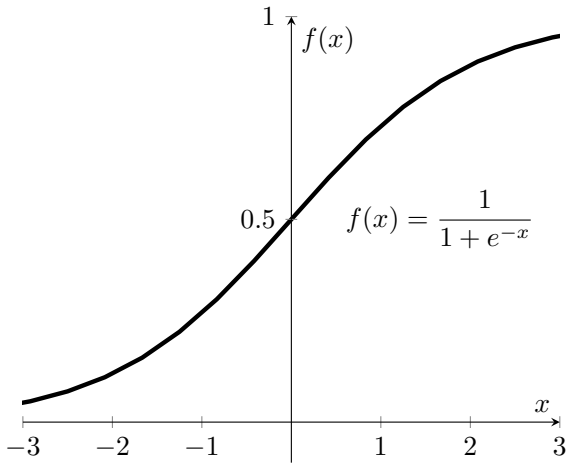


Figure 4.6: The sigmoidal (logistic) function.

Another reason for the use of sigmoidal functions is the fact that input values that are close to 0 can be better separated by the slope in this segment. Moreover, the absolute size of large positive or negative values, through in these areas flat curve, is relatively insignificant [127].

4.4.2 Perceptron

The perceptron is the most basic type of a neural network [128], [129]. An illustration of a simple perceptron is given in figure 4.7. It implements an artificial neuron with a threshold function whose incoming weights are adaptive and learned from data (cf. 4.4.1). The perceptron's output is a weighted linear combination of the inputs forced to the binary values $\{0, 1\}$ by means of a threshold. In the following representation, a set of inputs is given and the learning objective is to be able to classify these inputs into two categories which are labeled 0 and 1. Mathematically, the perceptron is defined by the following equation below:

$$h_\theta(x) = \begin{cases} 1 & \text{if } w^T x + b > 0 \\ 0 & \text{else} \end{cases} \tag{4.5}$$

where x is the input vector, w is the weights vector, and b is a bias weight. After the weights are initialized randomly or with zero values, they are adapted to minimize the error between the predicted output $\hat{y} = h_\theta(x)$ and the target $y \in \{0, 1\}$ using the perception learning rule:

$$w \leftarrow w + \alpha(y - \hat{y}) \bullet x \tag{4.6}$$

where $\alpha > 0$ is the learning rate and $\bullet$ denotes the element wise product of two vectors. The perceptron learning rule is similar to the gradient descent learning rule (cf. section 2.4).

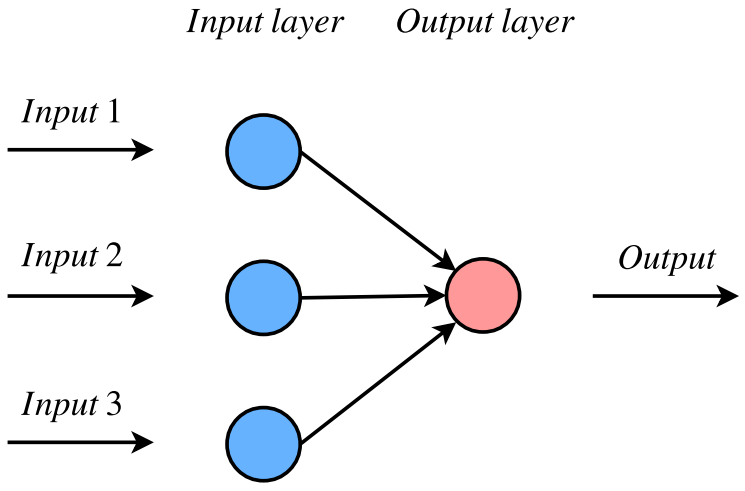


Figure 4.7: General perceptron model including one input and one output layer.

4.4.3 Multi-layer perceptron

The multi-layer perceptron MLP [130] is an extension of the perceptron consisting of multiple layers of neurons equipped with nonlinear activation functions. Contrary to the perceptron model the MLP is able to extract abstract representations of the inputs by stacking multiple layers of hidden neurons on top of each other. It has been proven [131–133] that the MLP is a universal function approximator, i.e. it can approximate any continuous function on a compact domain with arbitrary accuracy, provided it has at least a single hidden layer with a sufficient number of neurons. To formalize the model, let n_i denote the dimensionality of layer i in an MLP. Further, let $W^{(i)} \in \mathbb{R}^{n_i \times n_{i+1}}$ be the weight matrix from layer i to layer $i+1$, $b^{(i)} \in \mathbb{R}^{n_{i+1}}$ be the bias weight vector of layer $i+1$, and $\phi^{(i)} : \mathbb{R} \to \mathbb{R}$ is an element wise nonlinear activation function applied to layer i. Widely used activation functions are:

$$tanh(x) := \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{4.7}$$

$$logistic(x) := \frac{1}{1 + e^{-x}} \tag{4.8}$$

$$H(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases} \tag{4.9}$$

Figure 4.8 depicts commonly used activation functions with ANNs. A MLP with l layers including the input layer and output layer is described by the subsequent equations as follows:

$$z^{(1)} = x \tag{4.10a}$$

$$a^{(2)} = W^{(1)} z^{(1)} + b^{(1)} \tag{4.10b}$$

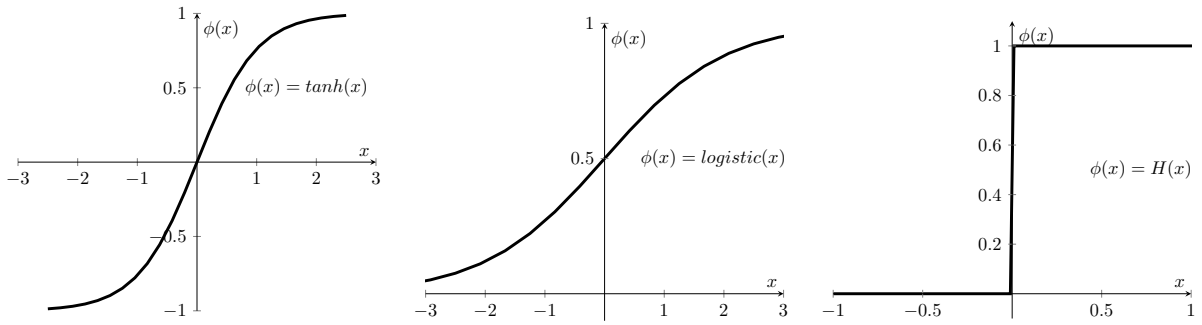
$$z^{(2)} = \phi^{(2)}(a^{(2)}) \tag{4.10c}$$

$$\vdots$$

$$a^{(l)} = W^{(l-1)} z^{(l-1)} + b^{(l-1)} \tag{4.10d}$$

$$z^{(l)} = \phi^{(l)}(a^{(l)}) \tag{4.10e}$$

$$\hat{y} = z^{(l)} \tag{4.10f}$$



(a) Hyperbolic tangent function. (b) Logistic function. (c) Heaviside step function.

Figure 4.8: Activation functions commonly in use with neural networks.

In figure 4.9 the MLP architecture is illustrated. Let $\theta := \{W^{(1)}, b^{(1)}, \ldots, W^{(l-1)}, b^{(l-1)}\}$ be the parameters of the given MLP formulas from 4.10a to 4.10d. Then, the optimal θ^* of the model are found by minimizing the error between the model output and the data $(x, y) \in D$ given the inputs x and the targets y.

$$\arg\min_{\theta} \frac{1}{|D|} \sum_{(x,y)\in D} \mathcal{L}(\hat{y}, y) \tag{4.11}$$

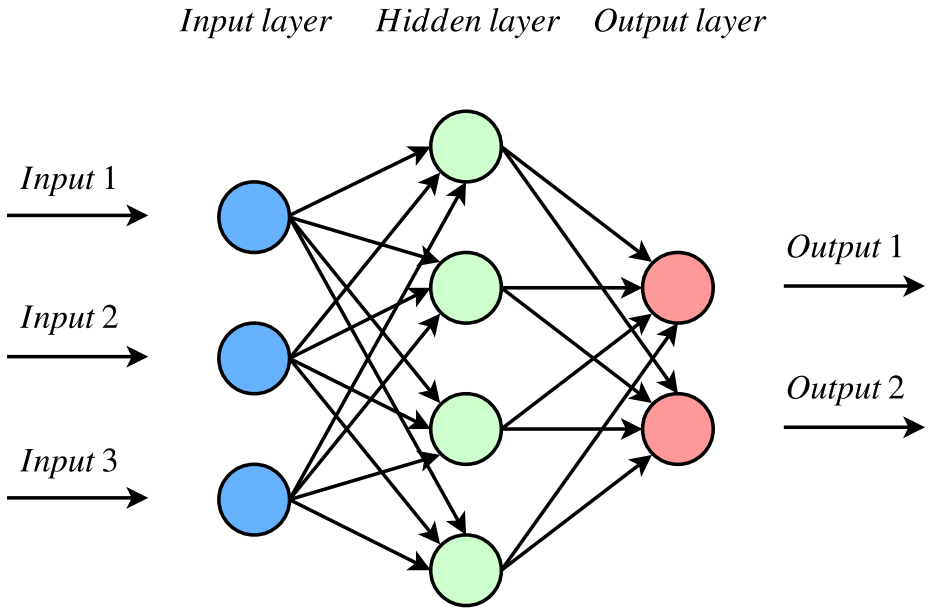


Figure 4.9: Multi-layer perceptron with l layers including the input, hidden and output layer.

4.4.4 Learning

The memory and the knowledge of an artificial neural network are in the weights of the compounds. The weight is then amplified when one neuron receives signals from another and both neurons are activated simultaneously [134]. This principle has been derived from [135]. The general form of the formula for the Hebbian theory is as follows:

$$\Delta w_{ij} = \eta x_i z_j \tag{4.12a}$$

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij} \tag{4.12b}$$

where neuron j receives input of neuron i. η is a constant learning rate, for example, in the interval between 0 and 1, which determines how much weight is to be changed in learning. The delta learning rule is an extension of the Hebbian learning rule. It is described as follows:

$$\Delta w_{ij} = \eta x_i(\overline{y}_j - y_j) = \eta x_i \delta_j \tag{4.13a}$$

$$w_{ij}(t+1) = w_{ij}(t) + \Delta w_{ij} \tag{4.13b}$$

where y_j is the actual output and $\overline{y}_j$ the expected output. It calculates the difference between the actual output and the desired output and the weights are adjusted accordingly. The learning rule notices that the learning process is over, which means the network has learned the predetermined data sufficiently, if the fault is kept to a minimum. The network error is the deviation of the actual value from the desired value for each output neuron. In an ANN the learning is usually followed by a certain sequence, based on a neural network learning algorithm, which consists, from a general point of view, of the following steps:

Algorithm 1 General steps in a neural network learning algorithm

```
1: repeat
2:     apply model to the network                                  ▷ Step 1
3:     if target output = actual output then                       ▷ Step 2a
4:         go to step 1
5:     else
6:         adjust weights of wrongly classified output neurons     ▷ Step 2b
7:     go back to step 1                                           ▷ Step 3
8: until
9: all patterns are recognized accordingly
```

During the learning procedure it may occur that connections or neurons are added/deleted, and parameter or function rules are changed respectively. Most frequently, however, the described weight change is concerned. Another important aspect of learning in the neural network is the training data. It is quite relevant to find the right balance between positive and negative training data. In this work, positive training data is considered as network data packages to be identified as normal. Negative training data is data of abnormal behavior, in which the network is taught to identify what is described as "normal state". Negative training data is necessary because otherwise the ANN shows a not acceptable learning rate.

Learning paradigms

An ANN has two different working principles, namely training and testing. The way, in which an ANN can learn from training data, can be classified into three types of learning during the training process (cf. [15], [68]). Anomaly detection techniques based on classification approaches can also be categorized based on the properties of the training data set. The training data set can either contain labeled instances from normal and abnormal classes, or only from one of the classes, typically from the normal class. Additionally, a training data set may contain unlabeled instances only. These statements lead to the following categorization:

- **Supervised learning** – In supervised learning both classes normal and abnormal are predetermined and labeled respectively. These classes can be conceived of as a finite data set consisting of input and output values. In practice, unseen data instances will be labeled with classes normal or abnormal.
- **Unsupervised learning** – Unsupervised learning is a task with regard to understanding and the discovery of hidden patterns in present data. In contrast to supervised learning, an ANN is not provided with predefined labels. Training data contains unlabeled instances only, and, assuming that regular instances are much more frequent than anomalies, the system's main task is to classify frequent data instances as normal classes.
- **Reinforcement learning** – In reinforcement learning the artificial system learns from a series of rewards or punishments and only the information whether the output class was correct or not is determined. The model is continuously improved based on processed data and the result.

4.4.5 Backpropagation

When reinforcement learning or supervised learning is used, it is advisable to combine these learning principles with backpropagation [136] as it is a common method for training an ANN [137]. The process for backpropagation is done by comparing the detected output of the network to the desired output and adjust the weights of the connections based on this difference [138]. This process starts with the output neurons and is performed in reverse learning direction and, if necessary, expanding to the input neurons. Learning with the backpropagation algorithm also involves problems. This is accompanied by the fact that backpropagation is referred to a "hill-climbing algorithm". If the error function is considered as a landscape of hills, then a specific location corresponds to a particular setting of connection weights. The height at this location is the value of the error function for these weights. The goal of the backpropagation algorithm is to find the place, where the error corresponds to the smallest value. Transferred to the landscape metaphor, the main goal is to find the deepest valley with the corresponding smallest error value. To note is that there are usually several local minima as can be seen in figure 4.10, of which only one represents a global minimum. If the algorithm has found a local minimum it can not decide whether a particular local minimum is also the desired global minimum. With this in mind, the shape of the error-prone surface is responsible for the success of the backpropagation algorithm.

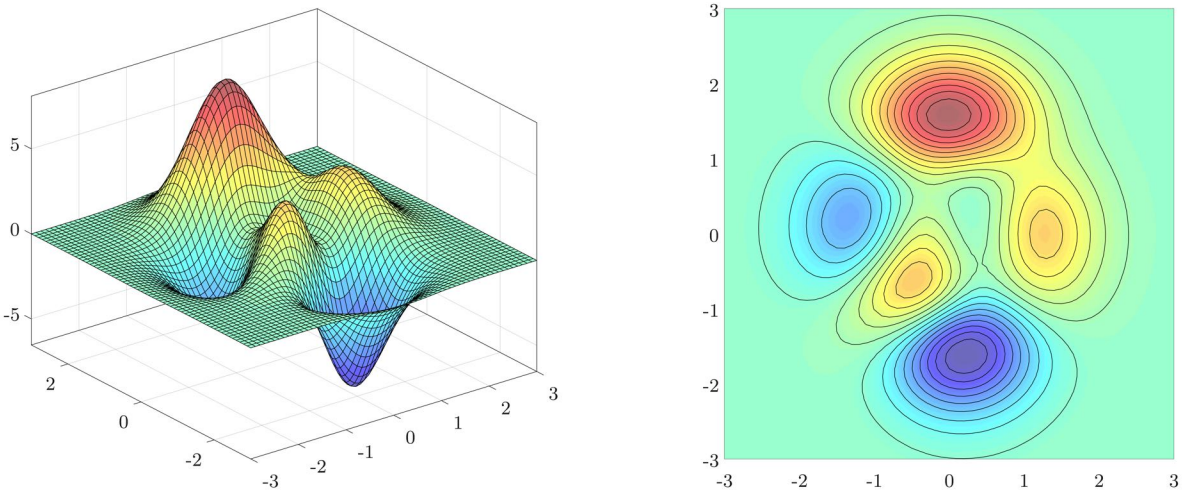


Figure 4.10: Surface (left) and contour plot (right) showing backpropagation algorithm using gradient descent with a learning rate of 0.1 and 100 iterations.

Part III

Application

Chapter 5

Anomaly detection in distributed systems

This chapter defines the problem solved by this thesis. In this regard, the starting point is given, preliminary works are described and facts from the previous chapter are employed to show conceptual differences.

This **thesis'** title "Anomaly-based intrusion detection for automotive networks" already hints at the more general approach of this thesis. For the actual use (i.e. the application to a real word problem), refinements are necessary.

The results are adapted and refined with the help of related research works and new approaches. Chapter 3 summarizes the common techniques used for anomaly detection and other concepts finding their way into this field of research. As mentioned before, the difference lies in the objectives which are defined by the class of systems which has to be operated and the group of anomalies, which has to be recognized based on particularly provided data. The main idea is to develop a system which is capable of recognizing anomalies by observing the message exchange between the participants of a CAN bus system. The information about the system, that is necessary to run the anomaly detection system, has to be minimal.

5.1 Objectives

The system consists of five main function blocks. The principal workflow can be described as follows: Based on a positive (cf. 4.4.4) data trace from the source, the learner derives the model. The source of the buffered CAN bus trace is called the original system. Emitted by the learner, the system model is then used by the detection system. It compares the actual behavior of the target system with the system model. Its output is a statement learner regarding the observer question whether the trace contains anomalies. Figure 5.1 depicts the principal structure of the anomaly detection system. The requirements and specifications concerning the function blocks in the figure below are discussed in the following sections.

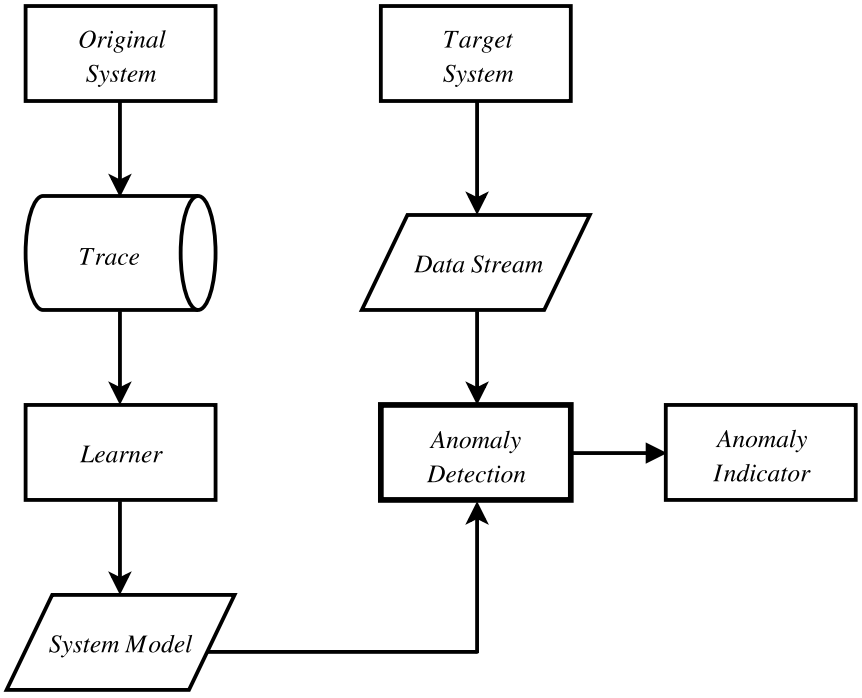


Figure 5.1: Outer framework of the anomaly detection system.

5.1.1 Anomalies

As previously mentioned in section 3.1, the term anomaly is understood as behavior deviating from regular test cases without malicious CAN data packages. The benchmark has to be stated by definition and memorized, so that it is possible to compare the actual system behavior with the behavior represented by the learned system model. Thus, it is necessary to learn and represent data that shows accurate operations of the system. An anomaly can have several different sources. As the system description used for the detection system is mapping uncritical behavior, an anomaly can be caused by uncritical system behavior which is not represented in the learned system; external influence on hardware, harmful as well as harmless; faulty behavior caused by a particular piece of software. A sophisticated question that could be answered, is which anomalies imply the existence of an intrusion and which type of intrusion in particular. This would be an important step towards a detection system, including functions for fault diagnosis as well as monitoring network traffic and protection against network and application-level attacks. However, this is an objective for further research projects, since an anomaly can be a symptom of a fault, i.e. the source is not directly derivable. Consequently, the problem is not intended to be solved at hand, however, it is considered as a future project towards an IPS and IDPS respectively. The following table 5.1 extends the definitions of the mathematical nomenclature, which can be helpful to describe the problem and the solution as well as to understand the relations.

Category	**Symbol**	**Members**		
Trace	$\mathcal{T}$ (Set)	Symbol		
Data stream	$\mathcal{D}$ (Set)	Symbol		
Symbol stream	$\mathcal{S}$ (Set)	Letter		
Alphabet	$\mathcal{A}$ (Set)	Letter		
Anomaly	$\vec{a}$ (Vector)	Symbol (exp./actual)	Time	
Anomalies	$\mathbf{A}$ (Matrix)	Symbol (exp./actual)	Time	
Fault	$\vec{f}$ (Vector)	Symbol (exp./actual)	Time	Automaton
Faults	$\mathbf{F}$ (Matrix)	Symbol (exp./actual)	Time	Automaton

Table 5.1: Symbol definition of mathematical nomenclature.

Concluding the statements above, the relationship between the different sets is as follows:

$$\mathcal{A} \subseteq \mathcal{S} \subseteq \mathcal{D} \subseteq \mathcal{T} \tag{5.1}$$

The task which has to be performed by an anomaly detection system is to determine whether there is an anomaly within the trace. A fault detection extends this function since it defines if one of the anomalies implies the existence of a fault. The dilemma resulting from the use case scenario and the system properties, is that only positive example traces can be used. When deriving the learned system from such a trace, all possible faults $\begin{pmatrix}1 & 1 & 0 & 0\end{pmatrix}^T$. $\mathbf{F}$ (i.e. the expected and the actual occurring symbol) are completely unknown. This is an important difference to most of anomaly detection methods that are described in chapter 3.

5.1.2 Target system

The target system has to be arranged in bus topology (cf. 2.2.2). As a direct consequence, the overall stream of messages can be accessed by the anomaly detection system, since all bus devices are using the same wire independent of source and sink. The original system has to be represented before the anomaly detection function is exerted, respectively. This thesis aims to enable the application of a detection system to a wide range of systems. However, the use case scenario analyzed is a state-of-the-art executive automobile (cf. 1.1.1).

5.1.3 Original system

As a basic assumption, the distributed system such as the target system (cf. section 5.1.2) can be described by the message traffic between its components (cf. [19]). When doing so, the states of the system do not stand for variables having a certain value, but rather for a system immanent status defined by the time (Δt), the traffic history, and the current message exchange. This approach differs from the common procedure, i.e. the one employed in other fields (cf. chapter 3), that involves the use of states representing an actual status of a system. In this case, the status can be described by variables exhibiting certain values, and thus can be identified by a human expert who analyzed the system behavior. If system states are defined by the properties described in the listing above, they might also be described by variables or other information accessible within the system. But it is also possible that some kind of hidden states are found, which cannot be derived from specification or documentation.

5.1.4 Subproblem

The objectives of this thesis lead to three main subproblems. Firstly, it is the essential question how the representation of the original system can be learned and stored efficiently. Considering the large extent of a real world system, the second task is to achieve a reduction of complexity. Thirdly, an integrated approach considering both issues has to be realized.

Learning of the original system

The first and elemental subproblem is the learning of the original system's representation. Since the states have to be formed by the data message stream observed, some further questions arise. A message stream is inherently an unidentified and continuous string of data. The used system substitutes have to show several properties which are declared below:

- **Correct, in a defined manner** – The representation has to last for a percentage that is necessary for a defined quality avoiding overfitting.
- **Verifiable** – Data access has to ensure the possibility to verify incoming data streams.

The terms *percentage*, *quality* and *overfitting* have to be clarified in relation to the topic in chapter 6. Necessary information regarding the original system has to be kept to a minimum.

5.1.5 Complexity reduction

When assuming complexity of a real distributed system to a high degree it has to be reduced from the beginning when the representative system properties are defined. An additional possibility is the partitioning of a large representation. This reduction is not intended to cause a loss of information, but rather to reduce redundancy. Due to the fact that the original system is cut into several sub-systems, the third sub-task arises. The sub-systems have to be represented and connected to account for the whole original system. The disadvantages with reference to the verification of the learned models have to be kept as small as possible.

5.2 Constraints

The detection system has to work in a network with a clearly arranged data stream. That is, there is no doubt about the order of the data. Since this thesis aims at developing a system model, resources are limited. Available time and memory are bounded so that a compressed representation of the system has to be adapted within a finite time frame with a memory occupation manageable for a contemporary embedded system. Another question that could be solved is to define which anomalies can be detected by the approach. Certainly, only those which have an impact on the message exchange can be recognized. The assumption made is that a significant amount of anomalies causes symptoms within the stream of data.

5.3 Data source

The available information regarding the system are:

- **Mask** – The mask enables the interpretation of the trace by masking the information with CAN identifier, timestamp, etc.
- **Trace** – The trace is a set of messages recorded in a defined timespan. All messages and the related time signature are known.
- **Status** – The information whether the trace is positive or not is essential. When a positive trace is considered as a negative one, the result is negated.

In this thesis, the testing and evaluation is conducted with CAN bus traces recorded in a standing or driving car. All of them are considered to be "positive", i.e. no behavior which implies the existence of an intruder or a failure could be observed. The definition of a "positive" trace is stated for this work as follows:

1. Warning bulbs are neither blinking nor glowing.
2. The automobile's behavior can be considered as predictable.
3. No obvious damages on the inside and the outside of the vehicle are visible.
4. The car is moved as it is reasonable in public traffic.

These definitions should be seen as a rule of thumb, as a more detailed definition would entail accurate analyses of every trace used. The required information can not be accessed.

5.4 Use case

With current topics such as autonomous driving and car connectivity, the E/E architecture of modern vehicles is fundamentally changing. The increasing networking with other traffic participants, back ends and passengers plays an essential role in this context. Technologies such as Wi-Fi, Bluetooth, vehicular telematics and Internet access expose the in-vehicle network to threats from external attacks. At the same time, the driving functions - especially with regard to autonomous driving - have to always work reliably. With this combination in mind, the automotive industry is facing major new and pressing challenges. A so-called “observer", which monitors the in-vehicle network communication, can make an important contribution to enhance the security of an automotive network. Hence, external attacks and internal malfunctions should be detected in order to take appropriate countermeasures. The concept and the implementation, which will be developed during the work on this thesis, have to be applicable to several automotive systems, although they are only tested with data, recorded from a stationary and moving car's internal network. Universality is not guaranteed since CAN packets and the underlying data varies from manufacturer to manufacturer. The limitations are stated in section 5.2. However, the use case scenario is a modern passenger car. The long-term direction is to extend this work towards an IDPS as an independent project, comprising leading car manufacturers. In this scientific field, there are some common definitions for anomalies that are likely to occur. Thus, it is possible to estimate a value for a detection rate that should be recognized. For a first prototype preprocessed system representations can be used. Nevertheless, for an application in a real environment, a live-learning approach (learning while the system is running), is necessary. Furthermore, no embedding or adaptation for the field application is compulsory. The anomaly detection system has to be developed for laboratory conditions, therefore the observer uses a preprocessed system model.

Part IV

Implementation

Chapter 6

Self-learning anomaly detection

This chapter presents and discusses the developed approach in order to realize a self-learning anomaly detection. Solutions for the considered subproblems described in the previous chapter are given and discussed.

To **allow** a good understanding of the coherence between the following subsections, the particular steps of the proposed detection approach are presented at first. In section 6.3 associated experiments from in-vehicle recordings are discussed and evaluated.

Officially released in 1986, the CAN is a serial bus system, developed primarily to protect a car against malfunctions and failures. When it comes to security, CAN provides no components against cyber attacks. Information security played only a subordinate role at the time of the development. Since the CAN bus is a broadcasting network, this nature may lead to maliciously injected messages performing various actions on different ECUs that might have a significant impact on driver and passenger safety. Replacing CAN with a more sophisticated bus may reduce risks, however, due to its widespread use and the time it would take until current models are scrapped, it is a primary aim to improve the security within the limitations of CAN. An important step towards this direction is the implementation of a system, detecting intruders on the in-vehicle network by analyzing the vehicular bus message traffic.

6.1 Concept

The basic idea of the anomaly detection approach is the representation of a normal (positive) state of a distributed vehicular system by a statistical model. This model has to be learned automatically by observing the participants' exchanging messages, in particular by observing a data stream without clearly defined beginning/ending. Arising subtasks are the compression of the derived model, the verification and the measurement of the model's quality.

6.1.1 System composition

The proposed system consists of preprocessing, statistical analysis and learning. These steps are part of the learning and absorption phase. Afterwards the comparison/emission phase takes place. As the concept is developed for an early prototype stadium (cf. section 1.1.1), an approach using a preprocessed model (i.e. the learning does not take place in the actual platform) is suggested. An overview of each subphase is given by the following itemization.

Absorption

- **Preprocessing** – The messages and submessages are categorized and the values used are calculated based on the system information. Furthermore, the symbols are defined and the data stream is created.
- **Timing Analysis** – The time distances between all the occurrences of the different symbols are measured and analyzed with respect to mean, variance and coefficient of variation.
- **Learning** – The neural network structure (cf. 6.2.4) learns the model.

Emission

- **Verification** – When the malicious CAN packet data is compared to a normal trace, differences have to be reported.
- **Quality assessment** – Using the metric, the quality of the learned automata is rated, i.e. the detection performance is measured.

6.1.2 Automaton type

The main intention is to learn a machine $\mathbf{a}_L$ representing the root system $\mathbf{a}_R$ and the original system $\mathbf{a}_O$ whose actual characteristics cannot be accessed completely without uncertainties.

- **Original system $\mathbf{a}_O$** – The original system is the actual system exerting the message exchange which is observed.
- **Root system $\mathbf{a}_R$** – The root system is part of the original system which is accessible, as it is mapped in the message exchange observed so far.
- **Learned system $\mathbf{a}_L$** – The learned system is the system represented by the learned machine.

Using the Angluin-style learner, a machine is learned that is defined by:

$$\mathbf{a}_L = (\mathcal{Z}_L, \mathcal{A}_L, \mathbf{T}_L) \tag{6.1}$$

$$\mathbf{T}_L = \begin{pmatrix} t_{11} & t_{12} & \cdots \\ t_{21} & \ddots & \\ \vdots & & t_{ij} \end{pmatrix};\ t_{ij} \in \{\mathcal{A}_\mathrm{L} \cup \{0\}\} \tag{6.2}$$

$$n_T = |\{t_{ij} | t_{ij} \in \mathcal{A}_L\}| \tag{6.3}$$

where $\mathcal{Z}_L$ is the number of learned states, $\mathcal{A}_\mathrm{L}$ is the alphabet and $\mathbf{T}_L$ is the transition matrix. All the transitions are deterministic (cf. 2.3), while states which define starting and end points within the automaton are not defined. Every state is considered as possible beginning and end point of a string. Nevertheless, there is at least one state with outgoing transitions only (excluding purely cyclic machines). The languages (cf. 2.3) essential for the proceeding are denoted as follows:

$$\mathcal{L}_{Automaton} = \mathcal{L}_A \text{ and } \mathcal{L}_{Stream\ of\ symbols} = \mathcal{L}_S \tag{6.4}$$

Due to the polynomial growth of the learning time, with regard to the system's complexity, a complete automotive system cannot be learned in an acceptable amount of time.

6.1.3 Categorizing sub messages

As described in the previous chapters, it is usually not feasible to model the original system $\mathbf{a}_O$ in the presumed problem scenario (cf. 6.1.2). The model of the the root system i.e. the model representing information contained in the message traffic, has to be as relevant as possible. This means that the used information has to represent the system behavior maximally well. The messages exchanged within the bus system contain discrete or continuous values. Certainly, all values are transmitted binary, but if they are treated the same way, the system's complexity increases dramatically. There is a simple reason to this: Continuous values still consist of the value transmitted as a binary number and the factor determining the real number. The assumption made at this point, is that the number of values, which can be adopted by the given factor, will be higher than the number of states.

6.1.4 Cyclic messages

On a CAN bus and other bus systems, messages are sent cyclically. Messages with a certain ID can be found on the bus with a defined distance in time within a defined tolerance. Figure 6.1 illustrates existing cycle times in CAN messages. The abscissa represents the amount of CAN IDs showing the same interval. The respective intervals are shown on the ordinate. There are more cycle times than the obvious ones such as 250ms, 500ms etc. and their multiples. Periodically sent variables, like checksums (depending on their definition) or system-related events, are mapped and interdependencies can also be recognized respectively.

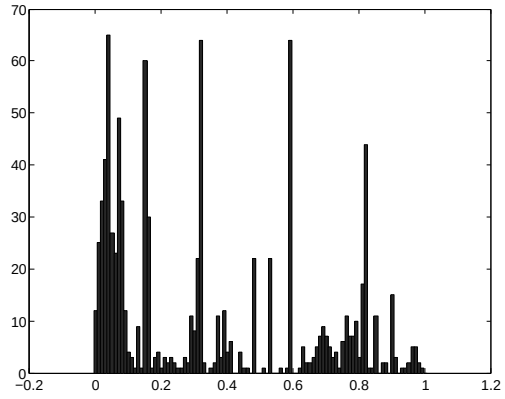


Figure 6.1: Histogram showing cyclic CAN messages for the used data set.

6.2 Training the neural network

During the detection phase, the learning mechanism of the proposed ANN structure is to classify normal and attack packet data. In this phase, the ANN learns to label classes correctly by adjusting its corresponding weights. Processing the training data step-by-step, the ANN makes use of the weights and transfer functions in the hidden layers and makes a comparison between the desired output and the actual output. Emerging errors are back-propagated to the network, causing the ANN to tweak the weights for the next input multiple times continually. Since the performance of an ANN is entirely dependent on the training performed, the identical training set is processed in a loop-like manner. The individual steps that are related to the iterative learning approach are presented and discussed subsequently.

6.2.1 Data acquisition

During the data acquisition phase, a moderately large amount of CAN bus traces from a modern car (cf. figure 6.2) was recorded using a CAN-to-USB interface, connected to the vehicle via on-board diagnostics interface. Also known as OBD-II and EOBD, the diagnostic interface is available in all petrol-powered cars made in 2001, and in 2004 or later for all diesel cars respectively [139]. OBD-II grants the reading and monitoring of emission-related signals such as the vehicle's speed, revolutions per minute and throttle position [140].

(a) Mercedes Benz E350 with 2987 ccm and 170 kW. (b) In-car OBD-II interface.

Figure 6.2: Test vehicle for recorded traces on a connected CAN bus using on-board diagnostics interface.

With regard to analyzing the CAN bus data, an open source PC-based monitoring software was chosen [141]. Figure 6.3 shows extracted raw CAN bus data from the used test vehicle.

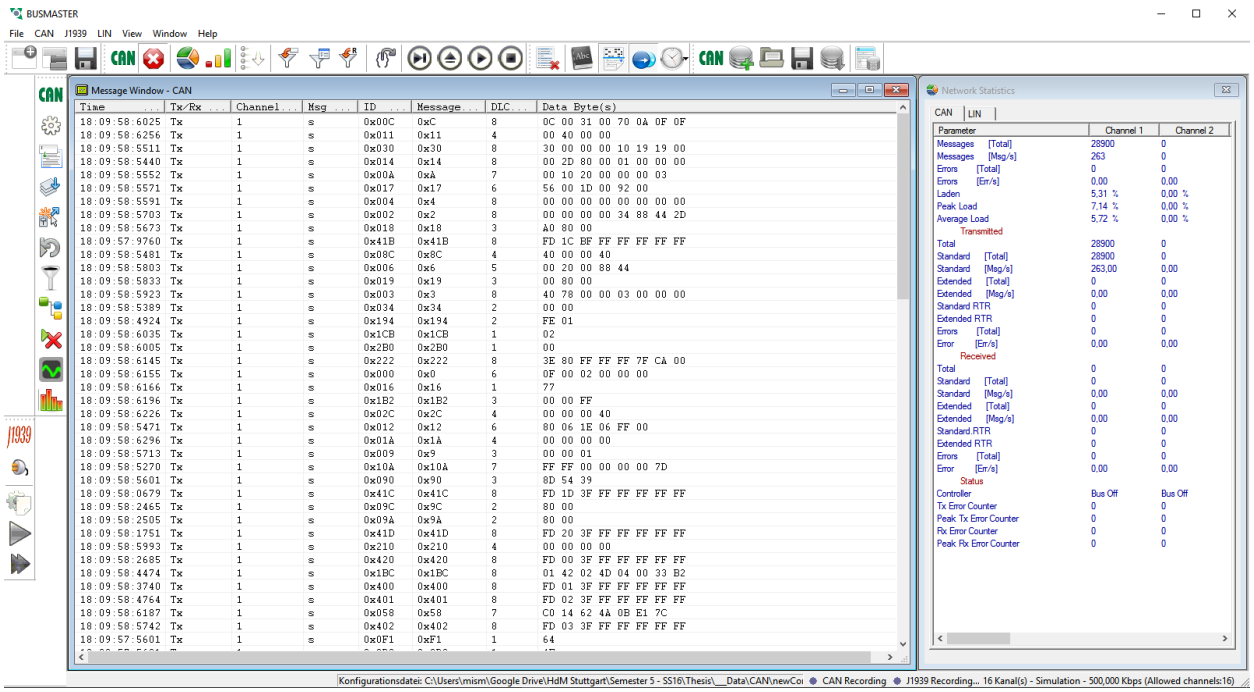


Figure 6.3: Open source analyzing software used for monitoring a raw CAN bus trace.

Although the CAN message payload data field may vary in length (maximum of 8 byte), messages with 8-byte-data were predominant. In-vehicle recordings comprise different scenarios, such as full braking, accelerating, sudden wheel steering and using the turn indicators randomly (cf. table 6.1). Five recordings, each with more than 800,000 messages are recorded using 80 kBit/s and 500 kBit/s access. Also, approximately 50 different IDs are identified.

Tool chain

The implementation is written in Python and compiled with JetBrain's PyCharm Professional 2016.1.3. Some parts of the statistical analysis are implemented using neural network toolbox for use with MathWork's MATLAB [142], particularly those for the feature extraction and learning. Due to the ease of use, preparation of recorded data is done with MathWork's MATLAB in version R2015b. Microsoft Windows 10 (x64) runs on the used test laptop with an Intel i5-2450M Dual Core processor with 2.5 GHz and 8 GB main memory.

Time	Bus ID	CAN ID	Received (Rx)	Payload
613.0132	1	222	Rx	0C 00 31 00 70 0A 0F 0F
613.0143	1	11	Rx	30 00 00 00 10 19 19 00
613.0166	1	30	Rx	FD 1C BF FF FF FF FF FF
613.0190	1	14	Rx	00 00 00 00 34 88 44 2D
613.0209	1	3	Rx	1E B3 31 1D F3 31 F2 CA
613.0277	1	17	Rx	08 16 00 00 88 00 02 57
613.0379	1	4	Rx	00 FF 21 81 FF FF 0E 52
613.0506	1	14	Rx	3C 00 01 19 17 19 19 00
613.0582	1	90	Rx	0C 08 39 00 70 0A 0F 0F
613.0600	1	12	Rx	8 3C 00 01 19 17 19 19 00

Table 6.1: Excerpt from a recorded CAN bus data trace.

6.2.2 Parameter extraction

Referring to an abstract representation of a CAN bus data packet, the CAN bus feature is mainly designed concerning computational efficiency. To put it differently, the CAN feature is extracted directly from a trace of CAN packets so that the extraction does not require decoding subsequently. Incidents of symbols in a data packet, particularly the *Data* field (cf. 2.2.2) that includes 8 byte (64) positions in the CAN bus syntax are taken into account. The following representation of the data vector can be mathematically described as follows:

$$P_o = \{P(b_i), ..., P(b_{63})\} \tag{6.5}$$

where $P(b_i)$ is the probability of the symbol "1" that is observed in the i-th position of a bit. Given a logistic function L: If $P(b_i)$ is < 0.5, the class is mapped to 0, if $P(b_i)$ is ≥ 0.5 it is depicted to 1. In order to generate the parameter, the entire bit positions comprised in the *Data* field may be of purpose, however, scaling down the dimension can be achieved by taking semantics, within a predefined range of an analogous syntax element, into account. The proposed technique regards a system model that takes the prevailing parameter (revolutions per minute, speed, throttle, accelerator pedal) acquired from the in-vehicle CAN where cross prediction is applied. An overview of extracted parameters is given in figure 6.4.

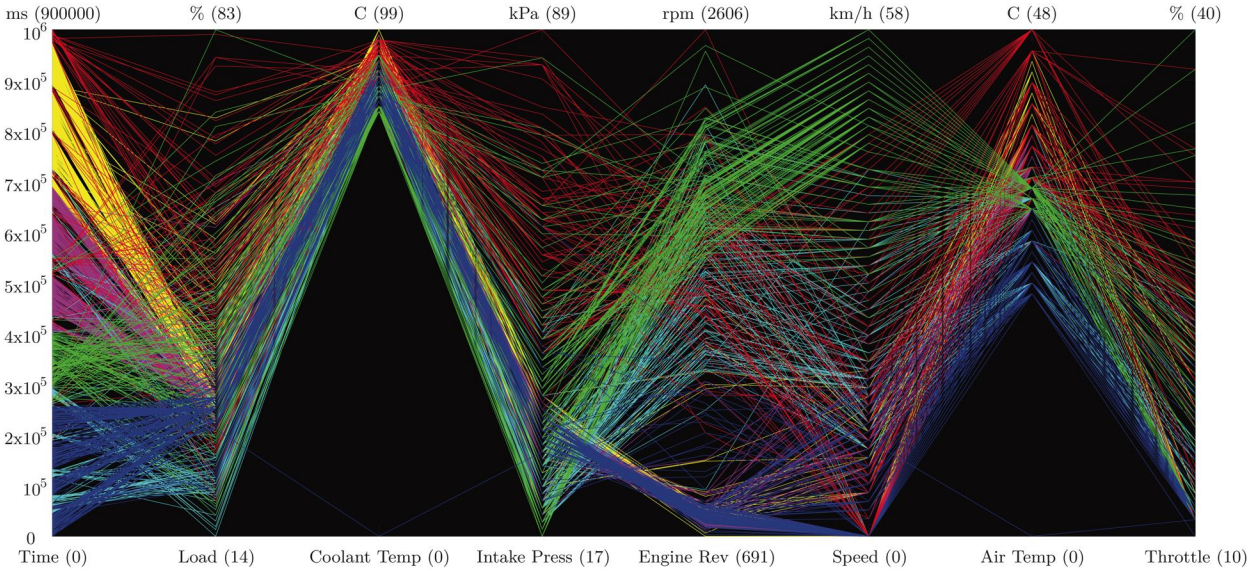


Figure 6.4: Manual parameter extraction from a test drive using a parallel coordinates plot with minimum values on bottom x-axis and maximum values on top x-axis.

6.2.3 Feature transformation

After analyzing and extracting the recorded data there are emerging problems to deal with:

1. The CAN packets carrying the parameters are sent at different frequencies, hence, the period of each parameter is not accurate because of the way CAN bus nodes contend for access.
2. The CAN ID also serves as a priority field, thus, the priority of the parameters are different among themselves.
3. The timestamps of associated data is not accurate, since the accuracy of the timestamps depends on the used hardware.

In order to solve the aforementioned problems before using the data packets to train a model, the parameters are normalized using parameter scaling and mean normalization techniques:

$$x_i := \frac{x_i - \mu_i}{s_i} \tag{6.6}$$

where μ_i is the average of all values for parameter i and s_i is the range of values (max - min), or the standard deviation respectively. Table 6.2 shows the preprocessed (subsampled) data.

Time	CAN ID	Load	Intake Press	Engine Rev	Speed	Throttle
240047.9	222	0.021255	0.000419287	0.02149	0.023041	0.021255
240047.9	11	0.021255	0.000419287	0.02149	0.023041	0.021255
240051.9	30	0.021255	0.000419287	0.02149	0.023041	0.021255
240052.9	14	0.021255	0.000419287	0.02149	0.023041	0.021255
240060.9	3	0.021255	0.000419287	0.02149	0.023041	0.021255
240061.9	17	0.021255	0.000419287	0.02149	0.023041	0.021255
240066.9	4	0.021883	0.000419287	0.02149	0.023041	0.021255
240073.9	14	0.021255	0.000419287	0.02149	0.023041	0.021255
240079.9	90	0.021255	0.000419287	0.02149	0.023041	0.021255

Table 6.2: CAN bus data packets after preprocessing using parameter scaling and normalization techniques.

6.2.4 Classification and learning

Figure 6.5 shows the ANN structure to classify both, normal and malicious CAN packet data.

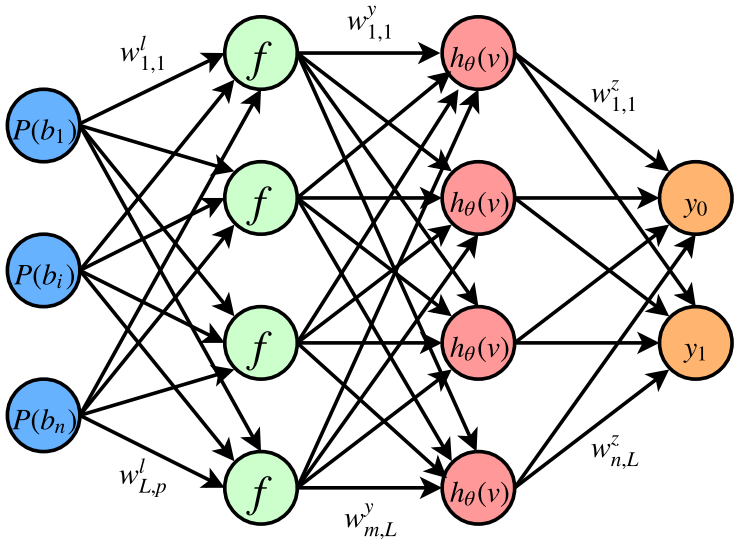


Figure 6.5: Artificial neural network structure in the proposed anomaly detection technique.

The structure comprises an input layer, several hidden layers and an output layer denoting normal and anomalous data packets. Each unit in figure 6.5 computes an output with a sigmoidal (logistic) activation function (cf. 4.4.1). Following this, the summed output vectors are transferred to subsequently hidden layers. Referring to a supervised learning problem (cf. 4.4.4) and the previous figure 6.5, a training set is given by $(v^1, y^1), (v^2, y^2), \ldots, (v^m, y^m)$ with m samples. Also, there is a feature vector v and y representing the binary label information, assigned to each training sample. While the learning phase is taking place, v denoting the input feature, moves through the perceptible input units at the beginning of the neural network structure, in which initial weights are given by the learning process 4.4.4. Following this, the weights are tweaked by hand. For this purpose, the cost function J denoting how well an ANN performs the mapping to correct output classes, given as the mean squared error between the predicted value and the output, is minimized by the subsequent equation:

$$J(w; v, y) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(v) - y\right)^2 \tag{6.7}$$

where w denotes the set of weights, y represents the label and $h_\theta(v)$ is the output function. Represented by J as the global cost function, a full batch training is given by the equation:

$$J(w) = \frac{1}{m} \sum_{i=1}^{m} J(w; v^i, y^i) + \frac{\lambda}{2m} \sum_{l=1}^{L} \sum_{i=1}^{S_l} \sum_{j=1}^{S_{l+1}} (w_{j,i}^{(l)})^2 \tag{6.8}$$

where L is the entire quantity of layers comprising the network, S_l is the amount of units (bias unit excluded) in the l-th layer, and $w_{j,i}^l \in w$ denoting the weight of the edges between a unit i within the layer $l-1$ and a unit j in the layer l. The goal is to obtain an optimal parameter set w^*, minimizing the error between the predicted output and the cost function:

$$w^* = \arg\min_w J(w) \tag{6.9}$$

this can be obtained by applying backpropagation (cf. 4.4.5) combined with gradient descent:

$$w_{j,i}^l := w_{j,i}^{l-1} + \alpha \frac{\partial}{\partial w_{j,i}^{l-1}} J(w) \tag{6.10}$$

where α represents the learning rate and controls how aggressively gradient descent is and $\frac{\partial}{\partial w_{j,i}^{l-1}}$ denoting a derivative term which updates the weight vector parameters simultaneously.

6.3 Testing and evaluation

This section evaluates the proposed detection approach regarding essential aspects. The performance as well as the quality of the developed and refined techniques are evaluated and discussed. With respect to this we refer to [143] where experiments on artificial neural networks were used for statistical anomaly intrusion detection for automotive CAN security.

6.3.1 Performance

For a simulation scenario 80 minutes of driving data and approximately 65000 records of CAN network traffic were collected. They were separated: 39000 records were used for training and 26000 for testing. The neural network was trained for 150 epochs (full training cycles applied on the prepared training set). The misclassification rate was measured as the percentage of inputs that were misclassified: false positives and false negatives (cf. figure 6.6 (red-colored)).

Figure 6.6: Confusion matrix for normal and malicious CAN packet data along with 64996 recorded data samples. 1170 examples of malicious data are misclassified as normal, and 1950 normal examples are incorrectly predicted as malicious. Overall, 95.2 % of the predictions are correct and 4.8 % are false predictions.

In figure 6.6, the first two diagonal cells denote the value and percentage of correct classifications by the trained network. 30743 data packets are correctly classified as normal. This corresponds to 47.3 % of all 64996 recorded samples. Similarly, 31133 packets are correctly classified as malicious. This corresponds to 47.9 % of all available data samples. 1170 of the malicious packets are incorrectly classified as normal and this corresponds to 1.8 % of all 64996 samples in the data set. Similarly, 1950 of the normal packets are incorrectly classified as malicious and this corresponds to 3.0 % of all data. Out of 31913 normal predictions, 96.2 % are correct and 3.7 % are wrong. Out of 33083 malicious predictions, 94.1 % are correct and 5.9 % are wrong. Out of 32693 normal cases, 94.1 % are correctly predicted as normal and 5.9 % are predicted as malicious. Out of 32303 malicious cases, 96.3 % are correctly classified as malicious and 3.7 % are classified as normal. Overall, 95.2 % of the predictions are correct and 4.8 % are wrong classifications with regard to 64996 available samples. The performance of the false negative and the false positive classification rates is acquired by using receiver operating characteristics. In figure 6.7, the concurrence between the false positive detection and the correct classification is represented using the following equation:

$$R_M\,(\%) = \frac{D_M}{T_M} \times 100 \tag{6.11}$$

$$R_N\,(\%) = \frac{D_N}{T_N} \times 100 \tag{6.12}$$

where R_M is the detection value of a malicious packet, R_N relates to the detection value of a benign packet, D_M is the value of detected malicious packets, D_N refers to the number of detected normal packets, T_M corresponds to the total number of malicious packets and T_N represents the total number of normal packets. By plotting the consolidation of the false positive value and the detection ratio with a given threshold, the curves in figure 6.7 can be achieved. It should be noted that a receiver operating characteristics curve shows an optimal classification performance when the data points are visualized more in the top-left corner. With respect to the performance progress of a neural network, a visual indicator at which the validation performance reached a minimum is helpful to determine if any changes need to be made to the network's architecture, the available data sets, or the training process. Figure 6.8 depicts the best performance, which is taken from the epoch with the lowest validation error. Determined by the cost function, the error reduces in general with increasing epochs.

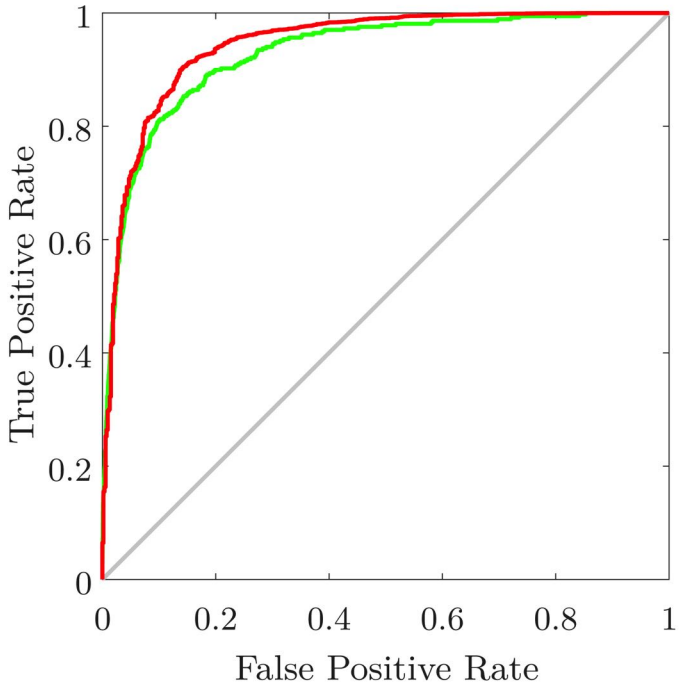


Figure 6.7: Receiver operating characteristics to measure the trade-off between the false positive detection and the correct classification for normal packets (green) and maliciously injected packets (red).

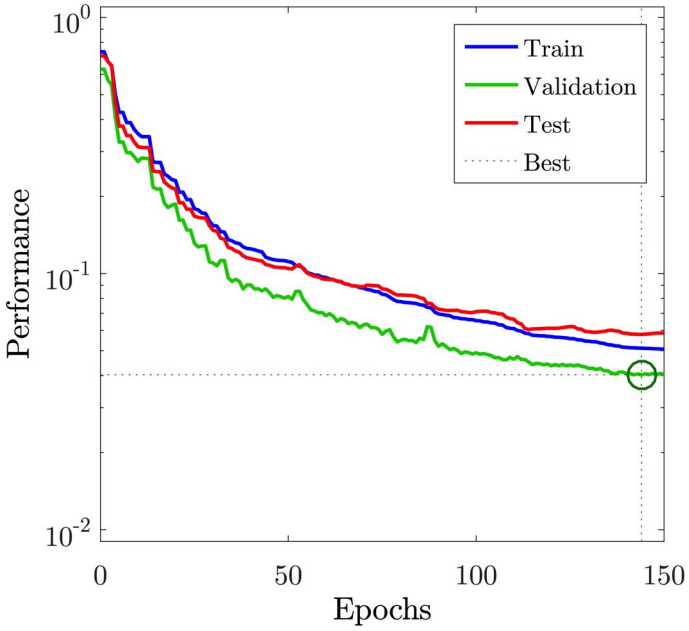
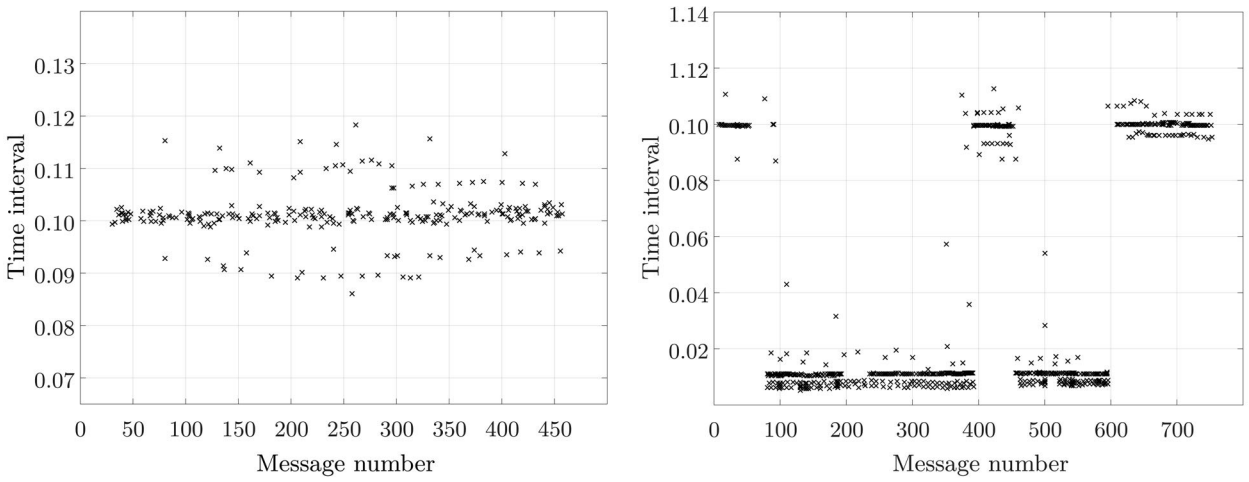


Figure 6.8: Best network performance, taken from the lowest validation error of 0.040244 at epoch 144.

6.3.2 CAN packet injection

There are two forms of CAN message injection strategies. On the one hand there is injecting CAN diagnostic messages, on the other hand is injecting standard messages to intimate the messages from ECUs [144], [145]. In general, diagnostic messages do not appear when a passenger vehicle is on the road. If a diagnostic message happens on the road, it is needed to be assumed that an attack or system malfunction may be the case [146]. During the evaluation experiments, message injection attacks are divided into two types for experiments. The first type covers the injection of specific CAN messages comprising a single CAN ID repeatedly to make the vehicle operate according to the injected messages [147]. Type two refers to injecting random or preordered messages of multiple CAN IDs to massively disrupting the CAN communication and eventually causing a system malfunction on the in-vehicle network [148], [149] or a denial of service [150], [151]. With regard to this, the vehicular CAN is simulated by sending data packets to the network that communicate with several ECUs on a broadcasting-based message transmission. CAN packets were generated using the Open Car Test-bed and Network Experiments (OCTANE) software and sent to the CAN via on-board diagnostics which typically provides direct access to one of the vehicle's CAN buses (modern cars contain multiple CANs). To address the problem of overfitting, a larger amount of generated packets were assigned to the training data than to the testing data. Attack scenarios comprise the injection of a single messages and multiple messages with various speed for making the CAN unavailable [152]. In the first place messages of a randomly selected single CAN ID with double, fivefold, and tenfold faster than origin cycle are injected. Generated messages are sent 10-100 times faster than the original ECU to make the target ECU listen to the injected messages (cf. [7]). Figure 6.9 represents the time intervals of selected CAN IDs at normal status and injection status, respectively. Messages are injected tenfold faster than the own cycle of the CAN ID, therefore, the time interval of injected messages is less than 10 % of the original interval. In the following example, the difference of time intervals between the normal status and malicious status can be clearly seen. In figure 6.9 each point represents an order and time interval of a message. The abscissa depicts the message generation number, whereas the ordinate is a time interval of a message. In figure 6.9b, the first message is generated at 0.055s and the second message is created at 0.157s, therefor, its point is marked at 25, 0.102.



(a) Normal CAN messages status.

(b) Malicious CAN messages status.

Figure 6.9: Scatter plot showing time intervals of CAN messages.

Further examination comprised injected messages of ten randomly selected CAN IDs with double, fivefold, and tenfold than original speed. In order to avoid the problem of overfitting (cf. section 4.2), injected messages are divided as follows: 30 malicious samples and 70 normal samples in double speed injection, 40 attack samples and 60 normal samples in fivefold speed injection, and 45 attack samples and 55 normal samples in tenfold speed injection. The experimental results of injected single and multiple messages are presented in table 6.3.

Message type	**Speed**	**Normal samples**	**Malicious samples**	**Performance**
Single CAN ID	Double	67	33	92.4 %
	Fivefold	52	48	93.1 %
	Tenfold	71	29	89.7 %
Multiple CAN IDs	Double	63	37	92.0 %
	Fivefold	74	26	95.6 %
	Tenfold	55	45	92.4 %

Table 6.3: Performance evaluation of experimental CAN message injection.

In summary, it can be established that there is a clear distinction between time intervals of messages under normal status and injected attack status. Time frames of a concretely chosen CAN ID in normal condition is approximately 0.10 seconds. In contrast to this, time intervals under injection attack conditions are nearly one-tenth of the normal time interval.

Chapter 7

Summary

The following chapter concludes the thesis, identifies the main contributions, highlights benefits and limitations of the proposed approach and discusses possible future works within the related field of research.

This **thesis** aims at providing a solution to the problem of detecting intruders in an automobile network by means of a self-learning mechanism. Arising tasks are related to the data acquisition the proposed learning technique including performance.

The principal motivation is to enable the use of formalized techniques to model a real-world distributed system by observing the message exchange of a modern passenger vehicle. In this regard, it is shown how a data stream can be abstracted by using statistical analysis. Based on this definition, a symbol stream is generated representing the message exchange that takes place between the bus participants. To access these information, basic bus-related data is required to allow the interpretation of the data stream. As the intention of the whole concept is to model a system with complexity to a high degree, it is proposed to segment the data stream resulting from the abstraction described above. In this case, an intense complexity means that the amount of occurring events is high in a certain period of time. To achieve the desired learning, an ANN is exerted using the results of a statistical analysis.

7.1 Contributions

It is shown how the approach of anomaly-based intrusion detection can be applied to learn a representation of a real system. For that purpose, CAN bus traces of a modern vehicle are used in order to introduce an event-based system representation. The particular steps are evaluated and analyzed to provide further development. In the end, it is demonstrated how the actual concept can be used for detecting anomalous states in vehicular networks. Section 6.3.1 shows the performance of the developed detection system. It is rated in two ways: Predominantly, the quality of the detection is reviewed as well as the concurrence between incorrect positive detection and the proper classification is shown. Moreover, experiments that involve injected CAN packets are conducted. This analysis gives parameters indicating which events show similar behavior regarding quantity of occurrence, cycle time and the variation of the cycle time. The CAN traces defined during clustering are fed to a learning algorithm which derives automaton models. An ANN is employed for that purpose where parameters, extracted from the network, as probability-based feature vectors are trained. This is the basis of detecting anomalies emitted by a trace of a system's behavior.

7.2 Benefits

The proposed approach provides benefits in many ways. Anomaly detection offers advantages as part of IDS in general, but also certain improvements over signature-based techniques. A major benefit over signature-based IDS is that ANNs are capable of learn characteristics and patterns of unknown instances with increasing experience. The main advantage of the use of ANNs for intrusion detection is their flexibility. ANNs have the capability to monitor data within a vehicular network, even if the broadcasted CAN packets are incomplete or distorted. Both these characteristics are crucial in the context of an automobile network when it comes to receiving information with the provision of an unpredictable system failure. Since ANNs learn from historical data, predictions about future instances can be derived on the basis of the previously known behavior. ANNs can be used to recognize already known attacks with a very high detection rate and apply this knowledge to instances whose characteristics do not exactly match previously known attacks. As with the previous mentioned benefits, there are limitations of IDS in general and disadvantages which apply to anomaly-based IDS and in particular the possible use of ANNs within this field of research.

7.3 Limitations

Unlike data from the on-board diagnostics interface, converted raw CAN recordings from car manufacturers are highly confidential and available only as part of an ongoing project. Another serious drawback of anomaly detection systems is that they can only detect attacks that exhibit abnormal behavior. Also, the vulnerability during the learning phase poses a major problem. Ideally, attacks should be prevented during the system's learning phase. The dynamics of the learned network characteristics are also considered as an additional risks, since trained attack profiles from potential intruders may no longer seen as anomalous and, thus, as an attack. The further drawback of anomaly-based intrusion systems following, is that they are an expensive and maintenance-intensive approach. Expensive, since high demands towards the hardware and software, such as real-time capabilities and a sufficient memory space, are placed. Moreover, special employees are needed to initialize and maintain the detection system. With respect to maintenance-intensive, set threshold values have to be constantly updated and adapted after the initialization of the anomaly detection system. However, the high rate of false alarms (false positives), resulting from a narrowly trained detection system and a high rate of wrongly classified intruders (false negatives), resulting from a broadly trained algorithm, is probably the most obvious disadvantage of anomaly detection. When it comes to applying an ANN to the problem of intrusion or misuse detection, three primary arguments against this strategy arise. The first argument against ANNs refers to the requirements of the training itself. Since an ANN's ability to identify indications of possible intrusion is entirely dependent on the training of the system being monitored, the training data and the techniques are used rather critically. Secondly, in order to ensure that the results are statistically significant and correct, the training routine requires an enormous amount of data to be learned. The training of an ANN for the purpose of intrusion detection can require countless, individual attack sequences. The vast amount of this sensitive information is not only very hard to obtain but was also not available for this thesis. The most significant drawback of applying an ANN to IDS is the black box nature of ANNs. The user can neither specify an ANN what triggers a particular behavior, nor can he manually change the ANN to have a specific way of behaving. ANNs adapt their analysis in response to the training that is performed. The weights and transfer function are usually frozen after the ANN has achieved an acceptable success. Although the ANN is reaching an adequate probability of success, the basis for this degree of precision is not often known [153].

7.4 Outlook

Nowadays, most IDS work with signature-based analysis. Since both, anomaly-based and signature-based detection systems are still a long way away from being implemented as a standard for widespread use in mass-produced vehicles, in the opinion of the author, hybrid forms of different intrusion detection methods will be most likely to become established. Although many parts of this thesis work have been described in detailed, there are still research activities need to be made. If a possible attacker is able to send CAN packets on the CAN bus, this scenario might have a significant impact not only on the security of automobiles but also on driver and passenger safety. Technical implementations such as securing the CAN communication by cryptographic security mechanisms [154], functional and penetration testing of automotive components [155] or by evaluating anomaly detection in LIN, MOST, FlexRay or other vehicular bus systems [156], can not guarantee full compensation for cyber attacks on cars. In order to close the gap towards automotive security, the consideration of cyber attacks on cars must become an integral part of every design decision in terms of hardware and software. Car manufacturers as well as automotive technology suppliers should therefore consistently build up expertise from the field of conventional information security and apply given strategies as an integral part of their engineering change management.

Bibliography

[1] Flavio D. Garcia, David Oswald, Pierre Pavlidès, and Timo Kasper, "Lock It and Still Lose It—On the (In)Security of Automotive Remote Keyless Entry Systems," in *Proceedings of the 25th USENIX Security Symposium*. Austin, USA: IEEE, 2016, pp. 929–944.

[2] Charles A. Miller and Chris Valasek, "Adventures in Automotive Networks and Control Units," IOActive, Seattle, USA, Technical Report, 2014.

[3] ——, "Remote Exploitation of an Unaltered Passenger Vehicle," IOActive, Seattle, USA, Technical Report, 2015.

[4] Lotfi Ben Othmane, Harold Weffers, Mohd Murtadha Mohamad, and Marko Wolf, "A Survey of Security and Privacy in Connected Vehicles," in *Wireless Sensor and Mobile Ad-Hoc Networks*, Driss Benhaddou and Ala Al-Fuqaha, Eds. New York, USA: Springer New York, 2015, pp. 217–247.

[5] David Clare, Shane Fry, Helena Handschuh, Harsh Patil, Chris Poulin, Armin Wasicek, Rob Wood, David A Brown, Geoffrey Cooper, Ian Gilvarry, David Grawrock, Anand Rajan, Alan Tatourian, Ramnath Venugopalan, Claire Vishik, David Wheeler, and Meiyuan Zhao, "Automotive Security Best Practices: Recommendations for security and privacy in the era of the next-generation car," Intel Corporation, Santa Clara, USA, White Paper, 2015.

[6] Karl Koscher, Alexei Czeskis, Franziska Roesner, Shwetak Patel, Tadayoshi Kohno, Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage, "Experimental Security Analysis of a Modern Automobile," in *Proceedings of the 2010 IEEE Symposium on Security and Privacy*, ser. SP '10. Washington, D.C., USA: IEEE Computer Society, 2010, pp. 447–462.

[7] Charles A. Miller and Chris Valasek, "A Survey of Remote Automotive Attack Surfaces," *BlackHat USA*, 2014.

[8] Tong-Jin Park, Chang-Soo Han, and Sang-Ho Lee, "Development of the electronic control unit for the rack-actuating steer-by-wire using the hardware-in-the-loop simulation system," *Mechatronics*, vol. 15, no. 8, pp. 899–918, 2005.

[9] Shane Tuohy, Martin Glavin, Ciaran Hughes, Edward Jones, Mohan Trivedi, and Liam Kilmartin, "Intra-Vehicle Networks: A Review," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 2, pp. 534–545, 2015.

[10] Subir Biswas, Raymond Tatchikou, and Francois Dion, "Vehicle-to-Vehicle Wireless Communication Protocols for Enhancing Highway Traffic Safety," *IEEE Communications Magazine*, vol. 44, no. 1, pp. 74–82, 2006.

[11] Fan Yu, Dao-Fei Li, and David Crolla, "Integrated Vehicle Dynamics Control – State-of-the Art Review," in *Vehicle Power and Propulsion Conference, 2008. VPPC '08. IEEE*. Harbin, China: IEEE, 2008, pp. 1–6.

[12] Christof Ebert and Capers Jones, "Embedded Software: Facts, Figures, and Future," *Computer*, vol. 42, no. 4, pp. 42–52, 2009.

[13] Eugen Mayer, "Serial Bus Systems in the Automobile. Part 4. FlexRay for data exchange in safety-critical applications," Vector Informatik GmbH, Stuttgart, Germany, Technical Report, 2010.

[14] Hans-Hermann Braess and Ulrich Seiffert, Eds., *Handbook of Automotive Engineering*, ser. SAE-R. Warrendale, USA: SAE International, 2005, no. 312, oCLC: 723699011.

[15] Varun Chandola, Arindam Banerjee, and Vipin Kumar, "Anomaly Detection: A Survey," *ACM Computing Surveys*, vol. 41, no. 3, pp. 15:1–15:58, 2009.

[16] Michael Muter and Naim Asaj, "Entropy-Based Anomaly Detection for In-Vehicle Networks," in *Intelligent Vehicles Symposium (IV)*. Baden-Baden, Germany: IEEE, 2011, pp. 1110–1115.

[17] Moti Markovitz and Avishai Wool, "Field Classification, Modeling and Anomaly Detection in Unknown CAN Bus Networks." Cologne, Germany: ESCAR, 2015, pp. 1–10.

[18] Adrian Taylor, Nathalie Japkowicz, and Sylvain Leblanc, "Frequency-Based Anomaly Detection for the Automotive CAN bus," in *World Congress on Industrial Control Systems Security (WCICSS)*. Slough, United Kingdom: IEEE, 2015, pp. 45–49.

[19] Falk Langer, Dirk Eilers, and Rudi Knorr, "Fault Detection in Discrete Event Based Distributed Systems by Forecasting Message Sequences with Neural Networks," in *KI 2009: Advances in Artificial Intelligence*, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Doug Tygar, Moshe Y. Vardi, Gerhard Weikum, Bärbel Mertsching, Marcus Hund, and Zaheer Aziz, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, vol. 5803, pp. 411–418.

[20] Roland Schmitz and Walter Kriha, *Sichere Systeme: Konzepte, Architekturen Und Frameworks*, ser. Xpert.press. Berlin/Heidelberg, Germany: Springer-Verlag Berlin Heidelberg, 2009.

[21] Jittiwut Suwatthikul, "Fault detection and diagnosis for in-vehicle networks," in *Fault Detection*, Wei Zhang, Ed. Rijeka, Croatia: InTech, 2010.

[22] Jihene Rezgui and Soumaya Cherkaoui, "Detecting Faulty and Malicious Vehicles Using Rule-based Communications Data Mining," in *36th Conference on Local Computer Networks (LCN)*, Bonn, Germany, 2011, pp. 827–834.

[23] Christopher M. Bishop, *Neural Networks for Pattern Recognition.* New York, USA: Oxford University Press, Inc., 1995.

[24] Srinivas Mukkamala, Guadalupe Janoski, and Andrew Sung, "Intrusion Detection Using Neural Networks and Support Vector Machines." IEEE, 2002, pp. 1702–1707.

[25] Sandeep Nair Narayanan, Sudip Mittal, and Anupam Joshi, "Using Data Analytics to Detect Anomalous States in Vehicles," Tech. Rep., 2015.

[26] Norbert Schlingmann, "Diagnostic of off-highway machinery for agriculture," in *CTI Special Day "Diagnostics for Off-Highway Applications".* Lindau, Germany: Claas KGaA mbH, 2012.

[27] Richard Ree Brooks, Samuel Sander, Juan Deng, and Joachim Taiber, "Automotive System Security: Challenges and State-of-the-Art," in *Proceedings of the 4th Cyber Security and Information Intelligence Research Workshop (CSIIRW).* ACM Press, 2008, p. 1.

[28] Patrick Nisch, "Security Issues in Modern Automotive Systems," in *Secure Systems.* Stuttgart, Germany: Stuttgart Media University, 2011, pp. 1–6.

[29] Corey Thuen, "Commonalities in Vehicle Vulnerabilities," IOActive, Seattle, USA, Technical Report, 2016.

[30] Rainer Kallenbach, "Trends in Automotive Electronics," *Journal of Electrical Engineering*, vol. 7, no. 1, p. 92, 2007.

[31] Max Milbredt, "The Electronics and Microtechnology Industry in Germany," Berlin, Germany, Technical Report, 2015.

[32] Max Milbredt and Rico Trost, "The Automotive Electronics Industry in Germany: Connected, Electrified and Autonomous Cars," Berlin, Germany, Technical Report, 2015.

[33] Marko Kolbe and Jonathan Schoo, "The Automotive Electronics Industry in Germany," Berlin, Germany, Technical Report, 2014.

[34] Dorottya Papp, Zhendong Ma, and Levente Buttyan, "Embedded Systems Security: Threats, Vulnerabilities, and Attack Taxonomy," in *13th Annual Conference on Privacy, Security and Trust (PST)*. Izmir, Turkey: IEEE, 2015, pp. 145–152.

[35] Chris Blommendaal, "Information Security Risks for Car Manufacturers based on the In-Vehicle Network," Master's Thesis, University of Twente, Twente, Netherlands, 2015.

[36] Ivan Studnia, Vincent Nicomette, Eric Alata, Yves Deswarte, Mohamed Kaaniche, Youssef Laarouchi, and Matthieu Roy, "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*, Toulouse, France, 2013.

[37] Manfred Broy, Ingolf H. Kruger, Alexander Pretschner, and Christian Salzmann, "Engineering Automotive Software," *Proceedings of the IEEE*, vol. 95, no. 2, pp. 356–373, 2007.

[38] Catalin-Virgil Briciu and Ioan Filip, "The Challenge of Safety and Security in Automotive Systems," in *9th International Symposium on Applied Computational Intelligence and Informatics (SACI)*. Timisoara, Romania: IEEE, 2014, pp. 177–181.

[39] Christoph Marscholik and Peter Subke, *Road Vehicles - Diagnostic Communication*. Heidelberg, Germany: Hüthig Verlag, 2008.

[40] Daimler AG, "Diagram of electronic components in a car - CHM Revolution," Stuttgart, Germany, Technical Report, 2016.

[41] Robert Bosch GmbH, "CAN Specification Version 2.0," Stuttgart, Germany, Technical Report, 1991.

[42] Arjun Shrinath and Ali Emadi, "Electronic control units for automotive electrical power systems: Communication and networks," *Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering*, vol. 218, no. 11, pp. 1217–1230, 2004.

[43] Marko Wolf, André Weimerskirch, and Christof Paar, "Security in Automotive Bus Systems," in *Proceedings of the Workshop on Embedded Security in Cars (Escar)'04*, Bochum, Germany, 2004.

[44] ——, "Secure In-Vehicle Communication," in *Embedded Security in Cars*, Kerstin Lemke, Christof Paar, and Marko Wolf, Eds. Berlin/Heidelberg, Germany: Springer-Verlag, 2006, pp. 95–109.

[45] Dan Klinedinst and Christopher King, "On Board Diagnostics: Risks and Vulnerabilities of the Connected Vehicle," Carnegie Mellon University, Pittsburgh, USA, Technical Report, 2016.

[46] Hendrik Schweppe, "Security and privacy in automotive on-board networks," PhD Thesis, Telecom ParisTech, Paris, France, 2012.

[47] Eugen Mayer, "Serial Bus Systems in the Automobile. Part 5. MOST for transmission of multimedia data," Vector Informatik GmbH, Stuttgart, Germany, Technical Report, 2010.

[48] ——, "Serial Bus Systems in the Automobile. Part 3. Simple and cost-effective data exchange in the automobile with LIN," Vector Informatik GmbH, Stuttgart, Germany, Technical Report, 2010.

[49] ——, "Serial Bus Systems in the Automobile. Part 2. Reliable data exchange in the automobile with CAN," Vector Informatik GmbH, Stuttgart, Germany, Technical Report, 2010.

[50] Robert Davis, Alan Burns, Reinder Bril, and Johan Lukkien, "Controller Area Network (CAN) schedulability analysis: Refuted, revisited and revised," *Real-Time Systems*, vol. 35, no. 3, pp. 239–272, 2007.

[51] Tsutomu Matsumoto, Masato Hata, Masato Tanabe, Katsunari Yoshioka, and Kazuomi Oishi, "A Method of Preventing Unauthorized Data Transmission in Controller Area Network," in *75th Vehicular Technology Conference (VTC Spring)*. Yokohama, Japan: IEEE, 2012, pp. 1–5.

[52] Paul Carsten, Todd Andel, Mark Yampolskiy, and Jeffrey McDonald, "In-Vehicle Networks: Attacks, Vulnerabilities, and Proposed Solutions," in *Proceedings of the 10th Annual Cyber and Information Security Research Conference*, ser. CISR '15. New York, USA: ACM, 2015, pp. 1:1–1:8.

[53] "Bus Systems," in *Bosch Automotive Electrics and Automotive Electronics - Systems and Components, Networking and Hybrid Drive*, 5th ed., ser. Bosch Professional Automotive Information, Robert Bosch GmbH, Ed. Wiesbaden, Germany: Springer Fachmedien, 2014.

[54] Florian Hartwich, "CAN with Flexible Data-Rate Specification Version 1.0," in *Proceedings of the 13th International CAN Conference*. Hambach, Germany: Robert Bosch GmbH, 2012.

[55] Edward F. Moore, "Gedanken-Experiments on Sequential Machines," in *Automata Studies, Annals of Mathematical Studies*, vol. 34. Princeton, USA: Princeton University Press, 1956, pp. 129–153.

[56] Zvi Kohavi and Niraj K. Jha, *Switching and Finite Automata Theory*, 3rd ed. Cambridge, United Kingdom: Cambridge University Press, 2009.

[57] David Harel, "Statecharts: A Visual Formalism for Complex Systems," *Science of Computer Programming*, vol. 8, no. 3, pp. 231–274, 1987.

[58] Bela Bollobas, *Modern Graph Theory*, 1st ed. New York, USA: Springer New York, 2013.

[59] Zhiwei Gao, Carlo Cecati, and Steven X. Ding, "A Survey of Fault Diagnosis and Fault-Tolerant Techniques Part I: Fault Diagnosis with Model-Based and Signal-Based Approaches," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 6, pp. 3757–3767, 2015.

[60] Ehsan Moradi-Pari, Amin Tahmasbi-Sarvestani, and Yaser Pourmohammadi Fallah, "A Hybrid Systems Approach to Modeling Real-Time Situation-Awareness Component of Networked Crash Avoidance Systems," *IEEE Systems Journal*, vol. 10, no. 1, pp. 169–178, 2016.

[61] Mariagrazia Dotoli, Maria Pia Fanti, Agostino Marcello Mangini, and Walter Ukovich, "Fault Detection of Discrete Event Systems Using Petri Nets and Integer Linear Programming," *Automatica*, vol. 45, no. 11, pp. 2665–2672, 2009.

[62] Anuradha Kodali, Yilu Zhang, Chaitanya Sankavaram, Krishna Pattipati, and Mutasim Salman, "Fault Diagnosis in the Automotive Electric Power Generation and Storage System (EPGS)," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 6, pp. 1809–1818, 2013.

[63] Abraham Cherfi, Michel Leeman, Florent Meurville, and Antoine Rauzy, "Modeling automotive safety mechanisms: A Markovian approach," *Reliability Engineering & System Safety*, vol. 130, pp. 42–49, 2014.

[64] David M. Nicol, William H. Sanders, and Kishor S. Trivedi, "Model-Based Evaluation: From Dependability to Security," *IEEE Transactions on Dependable and Secure Computing*, vol. 1, no. 1, pp. 48–65, 2004.

[65] Ethem Alpaydin, *Introduction to Machine Learning (Adaptive Computation and Machine Learning)*, 3rd ed. Cambridge, United Kingdom: The MIT Press, 2014.

[66] Tenko Raykov and George A. Marcoulides, *An Introduction to Applied Multivariate Analysis*. New York, USA: Routledge Academic, 2008.

[67] Varun Chandola, "Anomaly Detection for Symbolic Sequences and Time Series Data," PhD Thesis, University of Minnesota, Minneapolis, USA, 2009.

[68] Victoria J. Hodge and Jim Austin, "A Survey of Outlier Detection Methodologies," *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, 2004.

[69] Eamonn Keogh, Jessica Lin, Sang-Hee Lee, and Helga Van Herle, "Finding the most unusual time series subsequence: Algorithms and applications," *Knowledge and Information Systems*, vol. 11, no. 1, pp. 1–27, 2006.

[70] Rüdiger W. Brause, Timm S. Langsdorf, and Hans-Michael Hepp, "Neural Data Mining for Credit Card Fraud Detection," in *Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, ser. ICTAI '99. Washington, D.C., USA: IEEE Computer Society, 1999, pp. 103–106.

[71] Weng-Keen Wong, Andrew W. Moore, Gregory F. Cooper, and Michael M. Wagner, "Bayesian Network Anomaly Pattern Detection for Disease Outbreaks," in *Proceedings of the 20th International Conference on Machine Learning (ICML)*, vol. 2. Washington, D.C., USA: AAAI Press, 2003, pp. 808–815.

[72] Yann Le Cun, Bernhard E. Boser, John S. Denker, Richard E. Howard, Wayne Habbard, Lawrence D. Jackel, and Don Henderson, "Handwritten Digit Recognition with a Back-Propagation Network," in *Advances in Neural Information Processing Systems 2*, David S. Touretzky, Ed. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1990, pp. 396–404.

[73] Simon Hawkins, Hongxing He, Graham Williams, and Rohan Baxter, "Outlier Detection Using Replicator Neural Networks," in *Data Warehousing and Knowledge Discovery*, Gerhard Goos, Juris Hartmanis, Jan van Leeuwen, Yahiko Kambayashi, Werner Winiwarter, and Masatoshi Arikawa, Eds. Berlin, Germany: Springer Berlin Heidelberg, 2002, vol. 2454, pp. 170–180.

[74] Andreas Theissler, "Detecting anomalies in multivariate time series from automotive systems," PhD Thesis, Brunel University, London, United Kingdom, 2013.

[75] ——, "Anomaly detection in recordings from in-vehicle networks," in *Big Data Applications and Principles (BIGDAP)*, Madrid, Spain, 2014.

[76] Rolf Isermann and Peter Ballé, "Trends in the application of model-based fault detection and diagnosis of technical processes," *Control Engineering Practice*, vol. 5, no. 5, pp. 709–719, 1997.

[77] Jiawei Han, Micheline Kamber, and Jian Pei, *Data Mining: Concepts and Techniques*, 3rd ed. Haryana, India; Burlington, USA: Elsevier Ltd, Oxford, 2011.

[78] Eamonn Keogh, Jessica Lin, and Ada Fu, "HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence." Houston, USA: IEEE Computer Society, 2005, pp. 226–233.

[79] Gunjan Mansingh, Lila Rao, Kweku-Muata Osei-Bryson, and Annette Mills, "Profiling internet banking users: A knowledge discovery in data mining process model based approach," *Information Systems Frontiers*, vol. 17, no. 1, pp. 193–215, 2015.

[80] Martin Sachenbacher and Peter Struss, "Task-dependent qualitative domain abstraction," *Artificial Intelligence*, vol. 162, no. 1-2, pp. 121–143, 2005.

[81] IEEE Computer Society, "IEEE Standard Glossary of Software Engineering Terminology," *IEEE Std 610.12-1990*, pp. 1–84, 1990.

[82] Rui Abreu, Alberto González, Peter Zoeteweij, and Arjan J. C. van Gemund, "Using Fault Screeners for Software Error Detection," in *Evaluation of Novel Approaches to Software Engineering*, Leszek Maciaszek, César González-Pérez, and Stefan Jablonski, Eds. Berlin/Heidelberg, Germany: Springer Berlin Heidelberg, 2010, vol. 69, pp. 60–74.

[83] Olga Grinchtein, Bengt Jonsson, and Martin Leucker, "Inference of Timed Transition Systems," *Electronic Notes in Theoretical Computer Science*, vol. 138, no. 3, pp. 87–99, 2005.

[84] ——, "Learning of Event-Recording Automata," *Theoretical Computer Science*, vol. 411, no. 47, pp. 4029–4054, 2010.

[85] Martin Leucker and Christian Schallhart, "A Brief Account of Runtime Verification," *The Journal of Logic and Algebraic Programming*, vol. 78, no. 5, pp. 293–303, 2009.

[86] Moez Krichen and Stavros Tripakis, "Conformance Testing for Real-Time Systems," *Formal Methods in System Design*, vol. 34, no. 3, pp. 238–304, 2009.

[87] Christopher M. Bishop, *Pattern Recognition and Machine Learning*. New York, USA: Springer Verlag, 2007.

[88] Herve Debar, Monique Becker, and Didier Siboni, "A Neural Network Component for an Intrusion Detection System," in *SP '92 Proceedings of the 1992 IEEE Symposium on Security and Privacy*. Oakland, USA: IEEE Computer Society Press, 1992, pp. 240–250.

[89] Shelly Palmer, *Data Science for the C-Suite*. New York, USA: Digital Living Press, 2015.

[90] Yousef El, Ahmed Toumanari, Anouar Bouirden, and Nadya El, "Intrusion Detection Techniques in Wireless Sensor Network using Data Mining Algorithms: Comparative Evaluation Based on Attacks Detection," *International Journal of Advanced Computer Science and Applications*, vol. 6, no. 9, pp. 164–172, 2015.

[91] Sridhar Ramaswamy, Rajeev Rastogi, and Kyuseok Shim, "Efficient Algorithms for Mining Outliers from Large Data Sets," *ACM SIGMOD Record*, vol. 29, no. 2, pp. 427–438, 2000.

[92] James F. Allen, "Maintaining Knowledge about Temporal Intervals," *Communications of the ACM*, vol. 26, no. 11, pp. 832–843, 1983.

[93] Frank Höppner, "Learning Dependencies in Multivariate Time Series," in *Workshop on Knowledge Discovery from Temporal- and Spatio-Temporal Data*. Lyon, France: Springer Science & Business Media, 2002, pp. 25–31.

[94] Theophano Mitsa, *Temporal Data Mining*. Boca Raton, USA: Chapman & Hall/CRC, 2010.

[95] Srivatsan Laxman and P. S. Sastry, "A Survey of Temporal Data Mining," *Sadhana*, vol. 31, no. 2, pp. 173–198, 2006.

[96] Rahul Khanna and Huaping Liu, "System Approach to Intrusion Detection Using Hidden Markov Model," in *Proceedings of the 2006 International Conference on Wireless Communications and Mobile Computing (IWCMC)*. Vancouver, Canada: ACM Press, 2006, p. 349.

[97] George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung, *Time Series Analysis: Forecasting and Control*, 5th ed. Hoboken, USA: John Wiley & Sons, 2015.

[98] Kan Deng, Andrew W. Moore, and Michael C. Nechyba, "Learning to Recognize Time Series: Combining ARMA models with Memory-based Learning," in *International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*. Pittsburgh, USA: IEEE Computer Society Press, 1997, pp. 246–251.

[99] Hichem Sedjelmaci, Sidi Mohammed Senouci, and Mosa Ali Abu-Rgheff, "An Efficient and Lightweight Intrusion Detection Mechanism for Service-Oriented Vehicular Networks," *IEEE Internet of Things Journal*, vol. 1, no. 6, pp. 570–577, 2014.

[100] Hichem Sedjelmaci, Tarek Bouali, and Sidi Mohammed Senouci, "Detection and Prevention From Misbehaving Intruders in Vehicular Networks," in *IEEE Global Communications Conference*. Austin, USA: IEEE, 2014, pp. 39–44.

[101] Hichem Sedjelmaci and Sidi Mohammed Senouci, "A new Intrusion Detection Framework for Vehicular Networks," in *IEEE International Conference on Communications (ICC)*. Sydney, Australia: IEEE, 2014, pp. 538–543.

[102] Richard A. Kemmerer and Giovanni Vigna, "Intrusion Detection: A Brief History and Overview," *Computer*, vol. 35, no. 4, pp. 27–30, 2002.

[103] Christian Vestlund, "Intrusion Detection Systems in Networked Embedded Systems," in *TDDD17 Information Security*. Linköping, Sweden: Linköping University, 2015.

[104] Tobias Hoppe, Stefan Kiltz, and Jana 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.

[105] Ulf E. Larson, Dennis K. Nilsson, and Erland Jonsson, "An Approach to Specification-Based Attack Detection for In-Vehicle Networks," in *Intelligent Vehicles Symposium (IV)*. Eindhoven, Netherlands: IEEE, 2008, pp. 220–225.

[106] Michael Muter, Andre Groll, and Felix C. Freiling, "A Structured Approach to Anomaly Detection for In-Vehicle Networks," in *Sixth International Conference on Information Assurance and Security (IAS)*. Atlanta, USA: IEEE, 2010, pp. 92–98.

[107] Constantinos Patsakis, Kleanthis Dellios, and Mélanie Bouroche, "Towards a distributed secure in-vehicle communication architecture for modern vehicles," *Computers & Security*, vol. 40, pp. 60–74, 2014.

[108] Samuel Woo, Hyo Jin Jo, and Dong Hoon Lee, "A Practical Wireless Attack on the Connected Car and Security Protocol for In-Vehicle CAN," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–14, 2014.

[109] Hung-Jen Liao, Chun-Hung Richard Lin, Ying-Chih Lin, and Kuang-Yuan Tung, "Intrusion detection system: A comprehensive review," *Journal of Network and Computer Applications*, vol. 36, no. 1, pp. 16–24, 2013.

[110] Chih-Fong Tsai, Yu-Feng Hsu, Chia-Ying Lin, and Wei-Yang Lin, "Intrusion detection by machine learning: A review," *Expert Systems with Applications*, vol. 36, no. 10, pp. 11 994–12 000, 2009.

[111] Lynn Margaret Batten, Reihaneh Safavi-Naini, David Hutchison, Takeo Kanade, Josef Kittler, Jon M. Kleinberg, Friedemann Mattern, John C. Mitchell, Moni Naor, Oscar Nierstrasz, C. Pandu Rangan, Bernhard Steffen, Madhu Sudan, Demetri Terzopoulos, Dough Tygar, Moshe Y. Vardi, and Gerhard Weikum, Eds., *Information Security and Privacy*, ser. Lecture Notes in Computer Science. Berlin, Germany: Springer Berlin Heidelberg, 2006, vol. 4058.

[112] Pedro García-Teodoro, Jesús E. Díaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez, "Anomaly-based network intrusion detection: Techniques, systems and challenges," *Computers & Security*, vol. 28, no. 1-2, pp. 18–28, 2009.

[113] Jeremy Frank, "Artificial Intelligence and Intrusion Detection: Current and Future Directions," in *Proceedings of the 17th National Computer Security Conference*, Baltimore, USA, 1994.

[114] David A. Forsyth and Jean Ponce, *Computer Vision: A Modern Approach*, 2nd ed. Cambridge, United Kingdom: Pearson Publishing Ltd, 2011.

[115] Christopher D. Manning and Hinrich Schütze, *Foundations of Statistical Natural Language Processing*. Cambridge, USA: MIT Press, 1999.

[116] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning, "A survey of robot learning from demonstration," *Robotics and Autonomous Systems*, vol. 57, no. 5, pp. 469–483, 2009.

[117] Arthur L. Samuel, "Some Studies in Machine Learning Using the Game of Checkers. II-Recent Progress," *IBM Journal of Research and Development*, vol. 44, no. 1.2, pp. 206–226, 2000.

[118] Sergey Brin and Lawrence Page, "The Anatomy of a Large-Scale Hypertextual Web Search Engine," *Computer Networks*, vol. 56, no. 18, pp. 3825–3833, 2012.

[119] Claude Sammut and Geoffrey I. Webb, Eds., *Encyclopedia of Machine Learning*, 1st ed. New York, USA: Springer US, 2010.

[120] Stuart J. Russell and Peter Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. New Jersey, USA: Prentice Hall International, 2010.

[121] Prasanta Chandra Mahalanobis, "On the Generalised Distance in Statistics," *Proceedings of the National Institute of Sciences (Calcutta)*, vol. 2, pp. 49–55, 1936.

[122] Trevor Hastie, Robert Tibshirani, and Jerome Friedman, *The Elements of Statistical Learning*, 2nd ed., ser. Springer Series in Statistics. New York, USA: Springer New York, 2009.

[123] Ben Kröse and Patrick van der Smagt, *An Introduction to Neural Networks*, 8th ed. Amsterdam/Oberpfaffenhofen: The University of Amsterdam, 1996.

[124] Judith E. Dayhoff, *Neural Network Architectures: An Introduction.* New York, USA: Van Nostrand Reinhold Co., 1990.

[125] Wolfram-Manfred Lippe, *Soft-Computing*, ser. eXamen.press. Berlin/Heidelberg, Germany: Springer Verlag, 2006.

[126] David H. von Seggern, *CRC Standard Curves and Surfaces with Mathematica*, 3rd ed., Daniel Zwillinger, Ed. Boca Raton, USA: CRC Press, 2016.

[127] Eberhard Schöneburg, Nikolaus Hansen, and Andreas Gawelczyk, *Neuronale Netzwerke. Einführung, Überblick Und Anwendungsmöglichkeiten.* Haar, Munich: Markt & Technik Verlag, 1990.

[128] Frank Rosenblatt, "The Perceptron: A Probabilistic Model for Information Storage and Organization in The Brain," *Psychological Review*, pp. 65–386, 1958.

[129] Marvin Minsky and Seymour A. Papert, *Perceptrons: An Introduction to Computational Geometry*, expanded ed. Cambridge, USA: The MIT Press, 1987.

[130] Simon Haykin, *Neural Networks: A Comprehensive Foundation*, 2nd ed. Upper Saddle River, USA: Prentice Hall PTR, 1998.

[131] George Cybenko, "Approximation by Superpositions of a Sigmoidal Function," *Mathematics of Control, Signals, and Systems*, vol. 2, no. 4, pp. 303–314, 1989.

[132] Kurt Hornik, Maxwell Stinchcombe, and Halbert White, "Multilayer Feedforward Networks are Universal Approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[133] Kurt Hornik, "Approximation Capabilities of Multilayer Feedforward Networks," *Neural Networks*, vol. 4, no. 2, pp. 251–257, 1991.

[134] Oliver Kramer, *Computational Intelligence*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[135] Robert G. Morris, "D.O. Hebb: The Organization of Behavior," *Brain Research Bulletin*, vol. 50, no. 5-6, p. 437, 1999.

[136] Martin T. Hagan, Howard B. Demuth, and Mark Beale, *Neural Network Design*. Boston, USA: PWS Publishing Co., 1996.

[137] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[138] Sushmita Mitra and Yoichi Hayashi, "Bioinformatics With Soft Computing," *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)*, vol. 36, no. 5, pp. 616–635, 2006.

[139] Stephen Checkoway, Damon McCoy, Brian Kantor, Danny Anderson, Hovav Shacham, and Stefan Savage, "Comprehensive Experimental Analyses of Automotive Attack Surfaces," in *Proceedings of the 20th USENIX Conference on Security*, ser. SEC'11, San Francisco, USA, 2011, p. 6.

[140] ISO, "ISO 15031-5:2015 - Road vehicles – Communication between vehicle and external equipment for emissions-related diagnostics – Part 5: Emissions-related diagnostic services," Vernier, Geneva, Technical Report, 2015.

[141] Robert Bosch Engineering and Business Solutions Limited, "BUSMASTER," Technical Report, 2011.

[142] Howard Demuth and Mark Beale, *Neural Network Toolbox For Use with Matlab*, 1993.

[143] Satoshi Otsuka, Tasuku Ishigooka, Yukihiko Oishi, and Kazuyoshi Sasazawa, "CAN Security: Cost-Effective Intrusion Detection for Real-Time Control Systems," in *SAE World Congress & Exhibition*. Detroit, USA: SAE International, 2014.

[144] Charles A. Miller and Chris Valasek, "Car Hacking: For Poories," IOActive, Seattle, USA, Technical Report, 2013.

[145] ——, "CAN Message Injection," IOActive, Seattle, USA, Technical Report, 2016.

[146] Stephanie Bayer, Thomas Enderle, Dennis-Kengo Oka, and Marko Wolf, "Automotive Security Testing – The Digital Crash Test," in *Energy Consumption and Autonomous Driving*, ser. Lecture Notes in Mobility, Jochen Langheim, Ed. Springer International Publishing, 2016, pp. 13–22.

[147] Tobias Hoppe, Stefan Kiltz, and Jana Dittmann, "Applying Intrusion Detection to Automotive IT – Early Insights and Remaining Challenges," *Journal of Information Assurance and Security (JIAS)*, vol. 4, no. 6, pp. 226–235, 2009.

[148] Frederic Stumpf, Christian Meves, Benjamin Weyl, and Marko Wolf, "A Security Architecture for Multipurpose ECUs in Vehicles," in *25. VDI/VW-Gemeinschaftstagung: Automotive Security*, Ingolstadt, Germany, 2009.

[149] Khattab M. Ali Alheeti, Anna Gruebler, and Klaus D. McDonald-Maier, "An Intrusion Detection System Against Malicious Attacks on the Communication Network of Driverless Cars," in *12th Annual IEEE Consumer Communications and Networking Conference (CCNC)*. Las Vegas, USA: IEEE, 2015, pp. 916–921.

[150] Pierre Kleberger, Tomas Olovsson, and Erland Jonsson, "Security aspects of the in-vehicle network in the connected car," in *Intelligent Vehicles Symposium (IV)*. Baden-Baden, Germany: IEEE, 2011, pp. 528–533.

[151] Pierre Kleberger, "A Structured Approach to Securing the Connected Car," Licentiate Thesis, Chalmers University of Technology, Gothenburg, Sweden, 2012.

[152] Chung-Wei Lin and Alberto Sangiovanni-Vincentelli, "Cyber-Security for the Controller Area Network (CAN) Communication Protocol," in *International Conference on Cyber Security (CyberSecurity)*. Kuala Lumpur, Malaysia: IEEE, 2012, pp. 1–7.

[153] LiMin Fu, "A Neural Network Model for Learning Rule-Based Systems," in *International Joint Conference on Neural Networks (IJCNN)*, vol. 1. Baltimore, USA: IEEE, 1992, pp. 343–348.

[154] Jennifer Ann Bruton, "Securing CAN Bus Communication: An Analysis of Cryptographic Approaches," PhD Thesis, National University of Ireland, Galway, Ireland, 2014.

[155] Stephanie Bayer, Thomas Enderle, Dennis Kengo Oka, and Marko Wolf, "Security Crash Test – Practical Security Evaluations of Automotive Onboard IT Components," in *Automotive - Safety & Security 2014 (2015), Sicherheit Und Zuverlässigkeit Für Automobile Informationstechnik*. Stuttgart, Germany: Gesellschaft für Informatik e. V. (GI), 2014, pp. 125–139.

[156] Daniel Fallstrand and Viktor Lindström, "Applicability analysis of intrusion detection and prevention in automotive systems," Master's Thesis, Chalmers University of Technology, Gothenburg, Sweden, 2015.