HAW HAMBURG

BACHELOR THESIS
Martin Sergey Scherreiks

Fusion verteilter Anomalieerkennungssysteme in Fahrzeugnetzwerken

FAKULTÄT TECHNIK UND INFORMATIK
Department Informatik

Faculty of Engineering and Computer Science
Department Computer Science

HOCHSCHULE FÜR ANGEWANDTE WISSENSCHAFTEN HAMBURG
Hamburg University of Applied Sciences

Martin Sergey Scherreiks

Fusion verteilter Anomalieerkennungssysteme in Fahrzeugnetzwerken

Bachelorarbeit eingereicht im Rahmen der Bachelorprüfung
im Studiengang Bachelor of Science Informatik Technischer Systeme
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer: Prof. Dr. Franz Korf
Zweitgutachter: Prof. Dr. Jan Sudeikat

Eingereicht am: 07. Mai 2025

Martin Sergey Scherreiks

Thema der Arbeit

Fusion verteilter Anomalieerkennungssysteme in Fahrzeugnetzwerken

Stichworte

Anomalieerkennung, Datenfusion, Fahrzeugnetzwerke

Kurzzusammenfassung

Neben Verfahren wie der Intrusion Detection spielt die Anomalieerkennung eine zentrale Rolle für die Sicherheit von Fahrzeugnetzwerken. In dieser Arbeit wurde ein Framework, aufbauend auf bestehenden Werkzeugen, entwickelt, welches die Leistung von Algorithmen und ihren Konfigurationsmöglichkeiten bei der Anomalieerkennung untersucht. Zu den Konfigurationen gehören sowohl die Feature-Generation, als auch die konkreten Algorithmus-Parameter. Die Leistung wurde dabei in einer kleinen Bandbreite von Anomalieszenarien evaluiert, um Reaktionen und Eignung der Algorithmen auf die gezielten Angriffe einschätzen zu können. Aufgrund des nichtlinearen Wachstums des Rechenaufwands mit der Anzahl untersuchter Parameter wurde ein weiteres Augenmerk auf die Parallelisierung und Optimierung der internen Prozesse gelegt, die den Ressourcenverbrauch der Anomalieerkennung erheblich senken konnten. Um die Leistung der Anomalieerkennung zu steigern und das Potential von Datenfusion im Kontext der Anomalieerkennung zu untersuchen, wurden mehrere Fusionsverfahren implementiert und evaluiert. Die Ergebnisse zeigen, dass eine geeignete Kombination aus gut parametrisierten Algorithmen und Fusionsstrategien eine zuverlässige Anomalieerkennung ermöglichen können. Zusätzlich konnte gezeigt werden, dass durch die Fusion Muster in den Daten verstärkt werden, die den Umgang mit Fehlalarmen durch ihre zeitliche Korrelation mit tatsächlich stattfindenden Anomalien erheblich erleichtern. Verbesserungspotentiale bestehen jedoch bei der Identifizierung von Anomalie-Datenströmen, da auch nicht direkt manipulierte Datenströme Anzeichen von anormalem Verhalten zeigen. Die Erkenntnisse aus dieser Arbeit können dabei als Grundlage für weitere Forschung oder die Entwicklung von Anomaliedetektoren für den realen Einsatz dienen.

Martin Sergey Scherreiks

Title of Thesis

Fusion of Distributed Anomaly Detection Systems in Vehicular Networks

Keywords

Anomaly Detection, Datafusion, Vehicular Networks

Abstract

Besides methods such as intrusion detection, anomaly detection is an important measure to increase the security of vehicular networks. This thesis presents a framework built upon existing tools, which is capable of evaluating the performance of different algorithms and their configurations regarding feature generation and algorithm parameters. A small bandwidth of anomaly scenarios was used to test the performance and eligibility of the algorithms. Due to the nonlinear growth of computing demands with the number of investigated parameters, different measures regarding parallelization and optimization were employed. To further boost the performance of anomaly detection and explore the potential use of data fusion, several fusion methods were implemented and tested. The results show the possibility of reliable anomaly detection when done correctly. Additionally, it was demonstrated that the data fusion process reinforces preexisting patterns, which can be helpful when dealing with false positives due to the temporal correlation between false positives and actual anomalies. Possible improvements can be achieved by identifying anomalous data streams, as not directly manipulated data instances also show signs of anomalous behavior. The key takeaways can be used as a foundation to further investigate and implement a real-world anomaly detector.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungen

ADAS Advanced Driver Assistance System.

AE Autoencoder.

CFS Correlation-Based Feature Selection.

D-S Dempster-Shafer.

DDoS Distributed Denial of Service.

DMS Delay Manual Steer.

EAB Eliminate Auto Brake.

EE Elliptic Envelope.

FN False Negative.

FP False Positive.

FPR False Positive Rate.

HBO Histogram-based Outlier.

HBOS Histogram-based Outlier Score.

HIDS Host-based Intrusion Detection System.

IDS Intrusion Detection System.

IF Isolation Forest.

IL Inject Lidar.

IVN In-Vehicle Network.

KM K-Means.

LIDAR Light imaging, detection and ranging.

ML Maschine Learning.

MLDS Multi-Level Dempster-Shafer.

MS Mean Shift.

NADS Network Anomaly Detection System.

NIDS Network-based Intrusion Detection System.

PCA Principal Component Analysis.

PCAPNG PCAP Next Generation Dump File Format.

QoE Quality of Experience.

QoS Quality of Service.

RC Reorder Camera.

SDN Software Defined Networking.

SVM Support Vector Machine.

TN True Negative.

TP True Positive.

TSN Time Sensitive Networking.

V2I Vehicle-to-Infrastructure.

V2R Vehicle-to-Roadside.

V2V Vehicle-2-Vehicle.

V2X Vehicle-to-Everything.

VANET Vehicular ad-hoc Network.

1 Einleitung

Die digitale Kommunikation existiert in der heutigen, technologisch weit fortgeschrittenen Welt überall und lässt sich nicht mehr wegdenken. Parallel zu den aufkommenden Möglichkeiten, die sich in vielen Domänen wie der Fahrzeugindustrie Schritt für Schritt etablieren, entwickeln sich stetig neue Angriffs- und Fehlerpotenziale. Diese realen Bedrohungen frühzeitig erkennen zu können, ist essenziell für effektive Gegenmaßnahmen und die Sicherheit im Straßenverkehr. Ungewollte Kommunikationsmuster manifestieren sich als Anomalien, also Abweichungen vom spezifizierten Verhalten, deren Auftreten nicht unbemerkt bleiben darf.

Ob es sich bei Anomalien nun um Angriffe oder Fehlfunktionen handelt – es ist wichtig, frühzeitig handeln zu können, um die Sicherheit des Fahrzeugs, des Fahrers und der Umwelt gewährleisten zu können. Um das Vertrauen in und die Akzeptanz von integrierten Netzwerküberwachungsmaßnahmen zu stärken, dürfen so wenig Fehler wie möglich passieren. Es gilt also, eine Möglichkeit zu finden, Fehlalarme zu vermeiden, ohne Anomalien unbemerkt zu lassen.

Diese Arbeit verfolgt das Ziel, Anomalien möglichst präzise und zuverlässig erkennen zu können. Dabei bewegt sich die Arbeit im Kontext einer simulierten Umgebung. Diese Simulation bietet die Möglichkeit, Netzwerkaufzeichnungen aus einem Fahrzeugnetzwerk zu erhalten, in denen verschiedene Arten von Anomalien (Eliminierung, Injection, Delaying und Reordering von Frames) auf bestimmte Datenströme angewendet werden können. Obwohl die Anomalien eine gewisse Bandbreite von Angriffen abdecken können, bieten sie keine vollständige Abdeckung. Unter den genannten Bedingungen sollen folgende Forschungsfragen und Problemstellungen untersucht werden:

- Sind die Algorithmen Elliptic Envelope (EE), Support Vector Machine (SVM), Autoencoder (AE), K-Means (KM), Mean Shift (MS), Histogram-based Outlier (HBO) und Isolation Forest (IF) im Kontext von Fahrzeugnetzwerken für den Einsatz in der Anomalieerkennung geeignet?

- Im Verlauf der Arbeit müssen große Datenmengen mit verschiedenen Verfahren und Konfigurationen verarbeitet werden. Welche Maßnahmen können getroffen werden, um eine effiziente Datenverarbeitung mit möglichst geringem Rechenaufwand zu ermöglichen?
- Bei der Anomalieerkennung in Fahrzeugen stellen Fehlalarme ein zentrales Problem dar. Je nachdem, in welchem Kontext die Fehlalarme auftreten, variiert auch ihre Kritikalität. Diese Kontextabhängigkeit erschwert eine einheitliche Betrachtung der Zuverlässigkeit von Anomaliedetektoren, sodass für eine differenzierte Betrachtung nicht nur die Anzahl von Fehlalarmen, sondern auch der Kontext des Auftretens untersucht werden muss.
- Inwieweit besitzt Datenfusion ein Verbesserungspotenzial für die Erkennungsleistung und Präzision im Kontext der Anomalieerkennung?

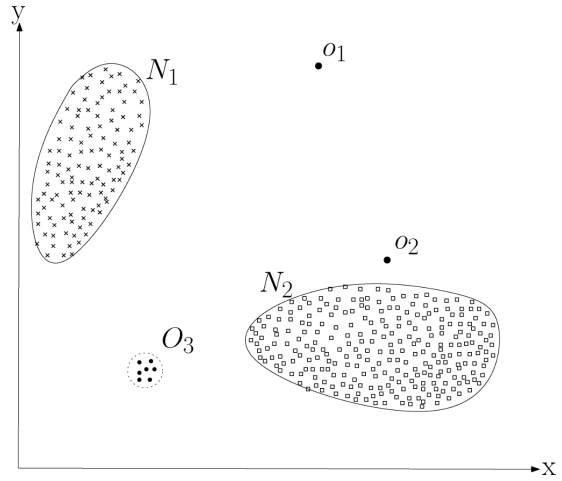
Um einen Einblick in das Thema zu geben sowie wichtige Konzepte und verwandte Arbeiten auf dem Forschungsgebiet vorzustellen, beginnt die Arbeit nach dieser kurzen Einleitung mit dem Grundlagenkapitel (Kapitel 2). Anschließend soll die Zielsetzung (Kapitel 3) die Rahmenbedingungen und Ziele der Arbeit konkretisieren. Im Anschluss beginnt der zweigeteilte Aufbau der eigentlichen Arbeit, in dem zunächst das Konzept (Kapitel 4) und die Umsetzung (Kapitel 5) der Anomalieerkennung vorgestellt werden. Die Ergebnisse der Anomalieerkennung sollen im darauffolgenden Kapitel (Kapitel 6) evaluiert werden, um Erkenntnisse für die Datenfusion zu sammeln und einen Vergleichswert zu schaffen. Danach folgt der gleiche Aufbau aus Konzept (Kapitel 7), Umsetzung (Kapitel 8) und Evaluation (Kapitel 9) für die Datenfusion. Mit den Ergebnissen aus den Evaluationen von Anomalieerkennung und Datenfusion sollen dann im Fazit (Kapitel 10) die Forschungsfragen beantwortet werden.

2 Grundlagen und verwandte Arbeiten

In diesem Kapitel wird ein Überblick über die für diese Arbeit relevanten Themen gegeben. Zudem werden Arbeiten vorgestellt, die sich bereits mit verwandten Fragestellungen befasst haben, um ein möglichst vollständiges Bild des aktuellen Stands der jeweiligen Forschungsgebiete zu vermitteln.
In der ersten Sektion wird zunächst der zentrale Begriff der Anomalie definiert und es werden verschiedene Kategorisierungen erläutert. Anschließend werden die Begriffe Software Defined Networking (SDN) und Time Sensitive Networking (TSN) in den gleichnamigen Abschnitten erklärt, da sie in der modernen Fahrzeugwelt eine wichtige Rolle spielen. Nach einer Übersicht über verschiedene Angriffsvektoren und -strategien folgt ein Blick auf mögliche Verteidigungsmaßnahmen. Abschließend werden die in dieser Arbeit verwendeten Algorithmen zur Anomalieerkennung, Fusionsverfahren sowie Evaluationsmetriken vorgestellt.

2.1 Anomalien

Anomalien sind Muster in Daten, die nicht der Definition eines normalen Verhaltens folgen. Sie werden daher nicht absolut, sondern nur im direkten Vergleich als solche klassifiziert. Das Beispiel in Abbildung 2.1 zeigt Regionen mit normalen Daten und Anomalien. In dieser erkennt man mit N1 und N2 Cluster normaler Daten, mit o1 und o2 einzeln liegende Anomalien und mit O3 ein kleines Anomalie-Cluster [vgl. 5, S. 1–2].
Allgemein lassen sich die zugrunde liegenden Datensätze durch die Anzahl ihrer unterschiedlichen Merkmale beschreiben und grundsätzlich je nach Dimensionalität in univariate und multivariate Datensätze einteilen. Univariate Datensätze besitzen eine, multivariate hingegen mehrere Dimensionen [vgl. 2, S. 308].
Es gibt verschiedene Gründe, warum Anomalien in Datensätzen auftreten können: Betrug, Infiltration und Terrorismus sind Beispiele für bösartige Angriffe. Auch Fehler in



Quelle: https://doi.org/10.1145/1541880.1541882 [5]

Abbildung 2.1: Die Abbildung zeigt ein einfaches Beispiel für Anomalien in einem zwei dimensionalen Datensatz.

Hard- und Software können jedoch Auslöser für Anomalien sein. Alle genannten Ursachen haben eine Gemeinsamkeit: Sie sind für die Analysten von großem Interesse, da unabhängig vom konkreten Auslöser Handlungsbedarf besteht. Dabei ist irrelevant, ob es darum geht einen Angriff zu stoppen oder einen Fehler zu beheben.
Die Erkennung von Anomalien steht in engem Zusammenhang mit der Rauschunterdrückung, da in beiden Fällen unerwünschte Dateninstanzen und Muster identifiziert und im Falle der Rauschunterdrückung entfernt werden sollen. Rauschen ist im Gegensatz zu Anomalien aus sicherheitstechnischer Sicht weniger relevant, spielt aber bei der allgemeinen Datenverarbeitung eine zentrale Rolle.
Ein weiteres verwandtes Thema ist die Neuheitserkennung: Dabei sollen Datenmuster erkannt werden, die nicht in den Trainingsdaten enthalten waren. Auch bei der Anomalieerkennung geht es darum, neben bekannten Angriffsmustern auch ungewöhnliche, neue Muster zu identifizieren, um auf bislang unbekannte Angriffe oder Systemfehler reagieren zu können.
Die Anomalieerkennung bringt verschiedene Herausforderungen mit sich, die den Einsatz komplexerer Ansätze erforderlich machen, als lediglich eine statische Region als „normal“ zu definieren: Häufig existieren keine klaren Grenzen zwischen normalen und abweichenden Daten. Zudem versuchen Angreifer gezielt, ihre Aktivitäten wie normales Verhalten erscheinen zu lassen. Auch verändern sich im Laufe der Zeit die Bereiche dessen, was als

normal gilt. Darüber hinaus gibt es keine einheitliche Definition von Normalität, die für alle Domänen gleichermaßen gilt.
Ein besonders schwieriger Aspekt ist die Beschaffung qualitativ hochwertiger Trainingsdaten, da dies mit erheblichem Aufwand verbunden ist. Ein bereits erwähnter Punkt betrifft zudem das Rauschen in den Daten, das sich teilweise nur schwer von Anomalien abgrenzen lässt. Aufgrund dieser Herausforderungen werden zur Anomalieerkennung in erster Linie Maschine Learning (ML)-Algorithmen oder neuronale Netze eingesetzt.

Anomalien lassen sich in drei grundlegende Kategorien einteilen:

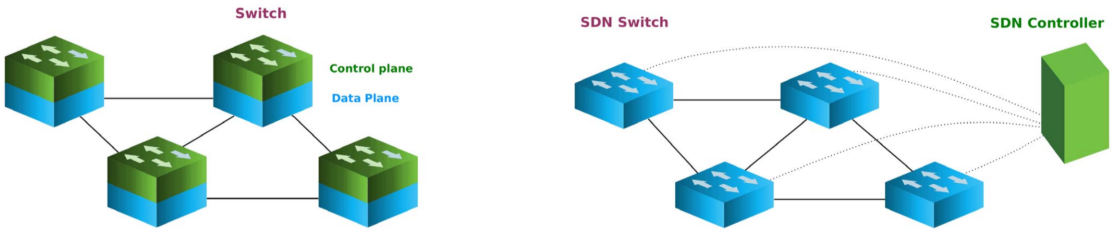
- **Punktanomalien** sind die einfachste Form von Anomalien, bei denen einzelne Dateninstanzen als abweichend klassifiziert werden. In Abbildung 2.1 sind o1 und o2 Beispiele für solche Punktanomalien, da sie außerhalb der Cluster und anderer Anomalien liegen.

- **Kontextuelle Anomalien** bezeichnen Dateninstanzen, die nur in einem bestimmten Kontext als Anomalie gelten, in anderen jedoch als normal. Der Kontextbegriff bezieht sich auf kontextuelle Attribute von Dateninstanzen, die deren Umgebung oder zeitliche Lage beschreiben. Zusätzlich gibt es verhaltensbezogene Attribute, die unabhängig vom Kontext sind. Zwei Instanzen mit denselben verhaltensbezogenen Attributen können somit je nach Kontext entweder als normal oder als Anomalie gelten.
  Die Effektivität von Verfahren zur Erkennung kontextueller Anomalien hängt maßgeblich davon ab, wie gut sich ein Kontext anhand der gegebenen Attribute definieren lässt. Solche Verfahren finden häufig in Zeitreihenanalysen Anwendung, bei denen beispielsweise Zeitpunkte als naheliegende kontextuelle Attribute dienen können.

- **Kollektive Anomalien** unterscheiden sich von den zuvor genannten Formen dadurch, dass sie nicht aus einzelnen, sondern aus Gruppen von Dateninstanzen bestehen. Die einzelnen Instanzen müssen dabei selbst keine Anomalien sein. Wie auch kontextuelle Anomalien treten kollektive Anomalien häufig in Zeitreihendaten auf. [vgl. 5, S. 1–2], [vgl. 2, S. 308]

2.2 Software Defined Networking

Mit den steigenden Anforderungen an moderne Netzwerke treten zunehmend strukturelle Schwächen in der bislang sehr starren Netzwerkinfrastruktur zutage. Insbesondere die grundlegend verteilte Architektur macht es bei Änderungen häufig erforderlich, zahlreiche einzelne Geräte manuell zu konfigurieren. Insgesamt lässt sich festhalten, dass es oft nur mit erheblichem Aufwand oder gar nicht möglich ist, zuverlässige Quality of Service (QoS)- oder Quality of Experience (QoE)-Garantien zu geben und einzuhalten.
QoS-Garantien betreffen dabei technische Aspekte der Datenübertragung, wie etwa die Zusicherung, dass ein Paket mindestens einmal ankommt. QoE-Garantien hingegen beziehen sich auf nutzerspezifische Anforderungen und die wahrgenommene Dienstqualität.



(a) Traditional architecture (b) SDN architecture

Quelle: https://doi.org/10.1109/COMST.2017.2782482 [1]

Abbildung 2.2: Die Abbildung zeigt den Vergleich zwischen traditionellen Overlay-Netzwerkarchitekturen und der SDN-Architektur.

Eine mögliche Lösung für die genannten Probleme und die Einhaltung entsprechender Garantien boten zunächst traditionelle Overlay-Netzwerke, die bereits vor den SDN-Lösungen existierten. Diese virtuellen Netzwerke bilden eine zusätzliche Schicht über der physischen Infrastruktur, gelten jedoch als komplex in Aufbau und Verwaltung.
Eine weniger komplexe und gleichzeitig flexiblere Lösung verspricht der Ansatz des SDN. Ein Vergleich der jeweiligen Architekturen ist in Abbildung 2.2 dargestellt. Der zentrale Unterschied liegt in der Trennung von Daten- und Kontrollebene. Dadurch übernehmen Switches lediglich die Aufgabe der Weiterleitung von Paketen, während die Kontrolllogik zentral in sogenannten SDN-Controllern verankert ist – wie auf der rechten Seite der Abbildung ersichtlich.
Diese Controller ermöglichen die programmatische Steuerung des Netzwerks. Neue Ver-

haltensweisen und Dienste können eingeführt werden, ohne dass dabei Änderungen an der zugrunde liegenden physischen Infrastruktur notwendig sind. Die Kontrollebene bildet die Schnittstelle zwischen programmatisch definierten Anforderungen und der Hardware-Ebene und ist verantwortlich für deren Umsetzung [vgl. 1, S. 333–338].

2.3 Time Sensitive Networking

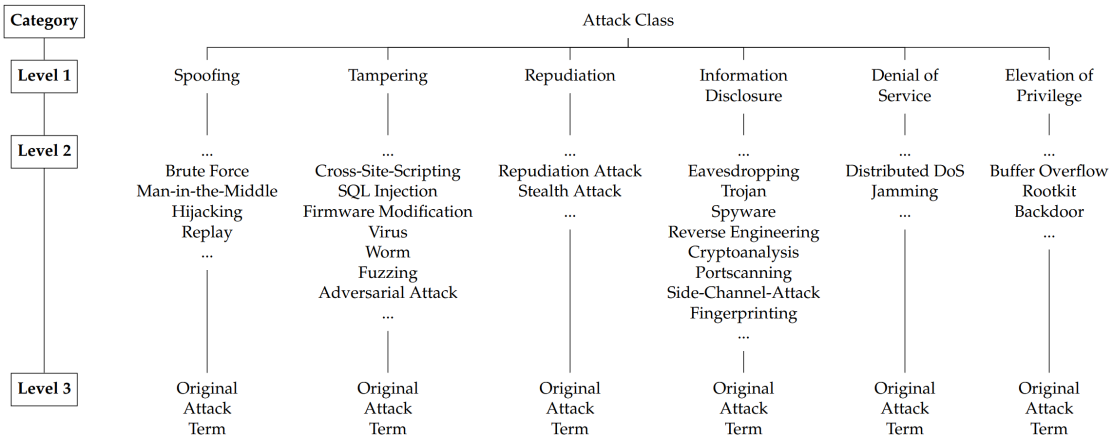
Ethernet ist das am häufigsten genutzte, universelle Protokoll, das viele Vorteile wie hohe Bandbreite, Skalierbarkeit und Kompatibilität besitzt. Da Ethernet in seiner Reinform allerdings Probleme mit den Echtzeitanforderungen im industriellen Umfeld hat, entwickelt die IEEE 802.1 Task Group den TSN-Standard, der sich gezielt auf die spezifischen Anforderungen in diesem Bereich fokussiert. Besonders für Fahrzeugnetzwerke ist die Unterstützung von Echtzeit-Datenverkehr von Bedeutung [17].

TSN ist eine Sammlung von Standards für QoS-Netzwerke. TSN ist also dafür verantwortlich, Zusicherungen machen und diese einhalten zu können. TSN organisiert den Datenverkehr in verschiedene Streams, die unterschiedlichen Klassen zugeordnet werden. Dabei kommen verschiedene Techniken zum Einsatz, um den unterschiedlichen QoS-Anforderungen gerecht zu werden. Synchrone und asynchrone Shaping-Techniken können so kombiniert werden, dass die für konkrete Anwendungen benötigten Datenverkehrsmuster realisiert werden können [19].

2.4 Angriffe auf Fahrzeugnetzwerke

Es gibt eine Vielzahl von Angriffsmöglichkeiten in Computernetzwerken. Eine in Sicherheitskreisen gängige Kategorisierung, die erstmals von Microsoft vorgestellt wurde und auch in der Arbeit von Sommer et al. [vgl. 26, S. 8] Verwendung findet, unterscheidet sechs Angriffsklassen: Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service und Elevation of Privilege, abgekürzt mit dem Akronym STRIDE. Darauf aufbauend können typische Angriffsmuster, wie Man-in-the-Middle-Angriffe, Viren oder Trojaner, diesen Kategorien zugeordnet werden. In Abbildung 2.3 lässt sich eine hierarchische Einordnung erkennen. Die umgekehrte Abstraktionshierarchie beginnt auf Level eins, auf dem sich die grundlegenden STRIDE-Begriffe befinden. Auf Level zwei, der

nächst niedrigeren Abstraktionsebene, werden diese spezifischer ausgeführt. Die konkreten Angriffe aus ihrem ursprünglichen Kontext werden auf Level drei abgebildet und in der Abbildung 2.3 stellvertretend als Öriginal Attack Term"bezeichnet.



Quelle: https://doi.org/10.3390/info10040148 [26]

Abbildung 2.3: Die Abbildung zeigt die hierarchische Kategorisierung von Angriffsmustern. Dabei stehen auf Level eins die grundlegenden Angriffsklassen nach dem STRIDE-Modell, die in Level zwei und drei weiter spezifiziert werden.

Neben dieser Abstraktionshierarchie existieren weitere Klassifizierungen von Angriffsmöglichkeiten, wie sie beispielsweise in der Arbeit von Talpur et al. [27] vorgestellt werden. Die Angriffe betreffen dabei nicht nur die In-Vehicle Network (IVN)s einzelner Fahrzeuge, sondern auch die Vehicle-2-Vehicle (V2V)-, Vehicle-to-Infrastructure (V2I)- und Vehicle-to-Roadside (V2R)-Kommunikation, die häufig unter dem Begriff Vehicle-to-Everything (V2X) zusammengefasst wird. Übergeordnet lassen sich Angriffe auf Vehicular ad-hoc Network (VANET)s in vier Klassen einteilen, die sowohl bei autonomen als auch nicht autonomen Fahrzeugen relevant sind: *Hardware/Software-based* (Angriffe auf Hardware-Komponenten und Softwaresysteme), *Infrastructure-based* (Angriffe auf Infrastrukturebene), *Sensor-based* (Angriffe auf Glaubwürdigkeit und Korrektheit von Sensoren wie dem Light imaging, detection and ranging (LIDAR)) sowie *Wireless Communication-based* (Angriffe auf die kabellose Kommunikation). Diese vier Klassen decken unterschiedliche Ebenen des Netzwerks ab, auf denen Angriffe erfolgen können.

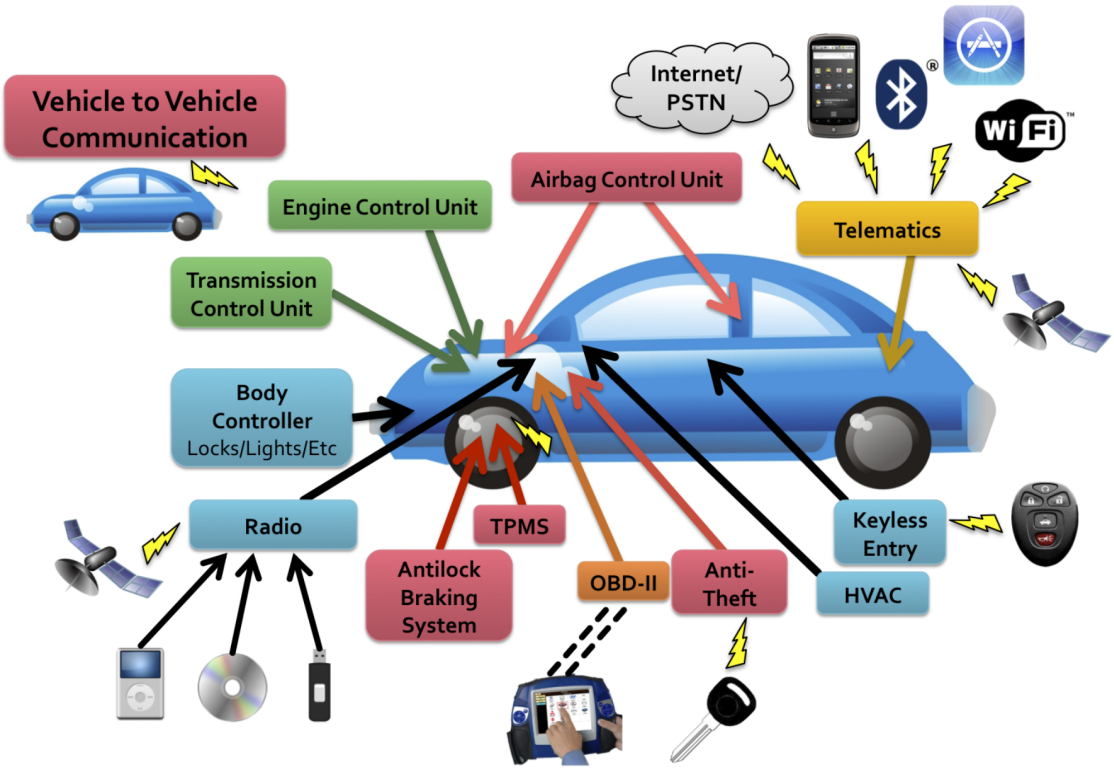
Die Angriffe innerhalb dieser Klassen zielen auf die Beeinträchtigung mindestens eines der vier Schutzziele ab: Verfügbarkeit, Vertraulichkeit, Integrität und Authentizität.

Die verschiedenen Einfallstore, die durch die Nutzung moderner Fahrzeugnetzwerke entstehen, lassen sich in Abbildung 2.4 erkennen und in drei Kategorien unterteilen [6].

- Die erste Kategorie bilden verschiedene direkte und indirekte **Physical Access Points (physische Zugänge)** zum Fahrzeugnetzwerk, wie z. B. der OBD-II-Port[1], der von Servicepersonal für die Diagnostik und Programmierung der Electronic Control Units ECUs verwendet wird. Bei älteren Fahrzeugen werden spezielle Geräte verwendet, während in modernen Fahrzeugen PCs zum Einsatz kommen, um via PassThru[2] auf den Port zuzugreifen. In beiden Fällen führt das dazu, dass sich Angreifer an den typischen Schwachstellen von PCs bedienen können, um Zugang zum Fahrzeugnetzwerk zu erhalten. Dazu zählen nicht nur die direkten Anschlüsse am Computer, sondern auch die Internetverbindung, die bei einigen Fahrzeugherstellern sogar Pflicht ist.
- In die zweite Kategorie fallen **Short-Range Wireless Access Points (kontaktlose Kurzstreckenzugänge)** zum Fahrzeugnetzwerk. Zu diesen Zugängen gehören bspw. Bluetooth und Wi-Fi des Autos oder der Remote Keyless Entry zum Öffnen der Türen. Unabhängig davon, welcher der Zugänge genutzt wird: Wenn es eine Sicherheitslücke in der ECU gibt (Hard- oder Softwareseitig), die den entsprechenden Zugang kontrolliert, kann ein Angreifer potentiell das Netzwerk kompromittieren.
- Die dritte und letzte Kategorie bilden die **Long-Range Wireless Access Points (kontaktlose Langstreckenzugänge)** zum Fahrzeugnetzwerk. Diese lassen sich in Broadcast- und adressierbare Kanäle unterteilen. Broadcast-Kanäle können nicht auf ein konkretes Fahrzeug gelenkt, aber vom Fahrzeug empfangen werden. Ein Beispiel für einen solchen Kanal stellen Satellitenradios dar. Adressierbare Kanäle auf der anderen Seite dienen dazu, mit konkreten Fahrzeugen zu kommunizieren. Viele Hersteller nutzen sogenannte Telematik-Systeme, welche mit einem Mobil-

[1] OBD-II-Port: Dieser fest verbaute Anschluss ermöglicht den physischen Zugriff auf die Software eines Autos.
[2] PassThru: Ein Verfahren, welches es ermöglicht, Daten zwischen einem Fahrzeug und externen Geräten auszutauschen.

funknetzwerk verbunden sind, um bspw. Diebstahlschutz oder Unfallmeldungen zu ermöglichen.



Quelle: https://www.usenix.org/conference/usenix-security-11/comprehensive-experimental-analyses-automotive-attack-surfaces [6]

Abbildung 2.4: Die Abbildung zeigt typische Angriffsziele auf ein Autonetzwerk.

2.5 Möglichkeiten der Verteidigung

Die Bandbreite an Angriffen auf ein Computernetzwerk ist groß. Um diesen nicht hilflos ausgeliefert zu sein, wurden und werden immer neue Verteidigungsmöglichkeiten entwickelt. Dabei beschäftigen sich viele Übersichtsarbeiten mit dem Vergleich zwischen verschiedenen Maßnahmen und haben die Ergebnisse verschiedener Arbeiten zusammengetragen und kategorisiert, von denen nun einige betrachtet werden sollen. In der Arbeit von Talpur et al. [27] werden verschiedene Klassen von Sicherheitslösungen vorgestellt,

die die Sicherheit des Fahrzeugs gewährleisten sollen und dabei auf ML zurückgreifen. Dabei geht es sowohl um die Sicherheit der IVNs als auch um die Sicherheit in der V2V-Kommunikation. Um bspw. den Diebstahl des Fahrzeugs zu verhindern, setzt die erste Klasse auf die Identifikation des Fahrers, ohne die Privatsphäre verletzen zu müssen. Dafür werden ML-Verfahren eingesetzt, um anhand des Fahrverhaltens ein Pseudonym zu erstellen. Die Lösungen der zweiten Klasse versuchen, spezifische Angriffe auf das Netzwerk zu erkennen. Zu diesen Angriffen gehören u. A. Distributed Denial of Service (DDoS)-Angriffe, die gerade gegen SDN-Netzwerke eingesetzt werden. Hierbei generieren Angreifer Anfragen an die SDN-Services, um den Controller außer Gefecht zu setzen. In der dritten Klasse finden sich die Lösungen, die Fehlverhalten und das Eindringen ins Fahrzeugnetzwerk erkennen sollen. Bei der sogenannten Intrusion Detection geht es darum, unbekannte Angriffsmuster zu erkennen. Die Lösungen der vierten Klasse stellen mit der sogenannten Trust Computation sicher, dass sich Fahrzeuge „ehrlich“ verhalten. In einer der Lösungen werden dafür in der V2V-Kommunikation Trust-Values für die Fahrzeuge erstellt. Diese bilden sich anhand von Informationen wie der Geschwindigkeit, dem Bremsstatus und der Position, die zwischen den Fahrzeugen ausgetauscht werden. Plötzliche Veränderungen in den Fahrzeuginformationen führen dabei zu einer Verschlechterung des Wertes. Zuletzt werden in der Arbeit die Lösungen vorgestellt, die die Privatsphäre sicherstellen sollen. Die Privatsphäre bezieht sich auf zwei Aspekte: User- und Location-Privacy. Wie schon bei den Lösungen für die Identifikation der Fahrer geht es hierbei vor allem um die Erstellung von Pseudonymen, um die Identität zu schützen.

Die Übersichtsarbeit von He et al. [15] hat sich explizit mit verschiedenen Verfahren für Intrusion Detection Systems (IDSs) befasst. Diese Systeme sind darauf spezialisiert, das Eindringen in ein Netzwerk wahrzunehmen. Allgemein werden die Systeme anhand ihres Einsatzortes in Network-based Intrusion Detection System (NIDS) und Host-based Intrusion Detection System (HIDS) klassifiziert. Wie die Namen vermuten lassen, beobachten NIDS das gesamte Netzwerk, in welchem sie eingesetzt werden, während die HIDS auf einem einzelnen Gerät (Host) im Netzwerk eingesetzt werden. Bei letzteren Systemen werden die Ressourcen des Geräts selbst beansprucht, und auch der Schutz bezieht sich nur auf das Einsatzgerät. NIDS besitzen den Vorteil, dass sie auch für größer skalierte Netzwerke effizient nutzbar sind und arbeiten typischerweise in drei Phasen: In der Monitoring-Phase werden verschiedene statistische Merkmale des Netzwerks analysiert. In der darauffolgenden Classification-Phase werden die extrahierten Merkmale genutzt, um sie mittels ML auf Zeichen eines potentiellen Angriffs zu untersuchen. Abhängig

von den Klassifizierungsergebnissen werden dann angemessene Verteidigungsmaßnahmen eingeleitet.
Neben der Einteilung nach dem Einsatzgebiet lassen sich die Arten von IDS auch anhand ihrer Methodik zur Erkennung von Angriffen klassifizieren. Signature-based NIDS betrachten die Payload (die Nutzlast) der Pakete im Netzwerk. Einzelne Sequenzen von Payload-Bytes werden dabei extrahiert und mittels Pattern-Matching-Algorithmen auf Angriffsmuster untersucht. Ein inhärentes Problem stellen unbekannte Angriffe dar, da nur nach bereits bekannten Signaturen gesucht werden kann. Auf der anderen Seite stehen anomaly-based NIDS, welche Merkmale aus dem gültigen Datenverkehr extrahieren und mit diesen ein „normales“ Profil für den Datenverkehr erstellen. Da sich diese Art der NIDS von vornherein nur auf das Lernen von normalen Mustern spezialisieren, besitzen sie auch eine natürliche Resistenz gegenüber sogenannten Zero-Day- und sonstigen unbekannten Angriffen.

Neben den Einsatzgebieten und den betrachteten Merkmalen im Netzwerk unterscheiden sich auch die verwendeten Lernmethoden. So gibt es einerseits die klassischen ML-Verfahren, wie bspw. Support Vector Machines, die auch in dieser Arbeit verwendet werden, und andererseits existieren Deep-Learning-basierte Verfahren. Einen besonderen Fokus legt die o.g. Übersichtsarbeit [15] auf sogenannte adversarial attacks. Diese Angriffe zielen insbesondere auf Deep Neural Networks ab. Sie arbeiten, indem sie für den Menschen kaum merkliche Veränderungen, sogenannte Perturbationen, an den Input-Daten vornehmen und damit eine andere Klassifizierung hervorrufen. Doch aufgrund der Anfälligkeit gegenüber dieser Art von Angriffen wurden auch Gegenmaßnahmen getroffen, die bspw. darauf abzielen, die internen Gewichte des neuronalen Netzes zu verstecken, um potentiellen Angreifern einen gezielten Angriff zu erschweren.

2.6 Algorithmen zur Anomalieerkennung

Es gibt verschiedenste Algorithmen, mit denen man Anomalien erkennen kann. Die hier vorgestellten Algorithmen bieten eine Bandbreite: von klassischen Klassifikatoren und Clustering-Algorithmen bis hin zu solchen, die auf Histogrammen oder Neuronalen Netzen basieren.

Support Vector Machines

Die SVM ist eine Entscheidungsmaschine. Sie ist darauf trainiert, Daten in verschiedene Kategorien einzuteilen. Dabei sind SVMs per se nur in der Lage, binäre Entscheidungen zu treffen. Die grundlegende Idee ist es hierbei, Daten, betrachtet als n-dimensionale Merkmalsvektoren, in eine von zwei Gruppen einzuteilen. Dabei wird die Annahme getroffen, dass die beiden Gruppen linear voneinander zu trennen sind. Es wird versucht, eine Grenze durch den Merkmalsraum zu legen, sodass alle Mitglieder der einen Gruppe auf einer Seite sind, während sich alle Mitglieder der anderen Gruppe auf der gegenüberliegenden Seite befinden. Da es unendlich viele Lösungen geben kann, wird die Grenze so gezogen, dass der Abstand der Grenze zu den Datenpunkten maximiert wird. Die Grenze definiert sich dabei durch die Support Vectors, also diejenigen Datenpunkte, die der Entscheidungsgrenze am nächsten liegen und damit ihren genauen Verlauf bestimmen. Auch wenn die Daten im linearen Merkmalsraum nicht linear separierbar sind, ist es durchaus möglich, die Daten in einem nichtlinearen Merkmalsraum linear zu separieren. Das bedeutet, die Daten werden mithilfe einer nichtlinearen Kernelfunktion so in einen höherdimensionalen Raum transformiert, dass sie linear separierbar sind. [vgl. 3, S. 325–331]

Elliptic Envelope

Der EE Algorithmus gehört zu den unsupervised learning Algorithmen. Der binär klassifizierende Algorithmus spannt eine elliptische Hülle um die Datenpunkte einer Klasse. Liegen klassifizierte Datenpunkte außerhalb dieser Hülle, gelten sie als Anomalie. Aufgrund seiner Funktionsweise eignet er sich besonders für Daten, die sich einer Gauß-Verteilung annähern. Außerdem gilt der Algorithmus als besonders resistent gegen Ausreißer und kann sowohl mit großen als auch hochdimensionalen Datensätzen umgehen. [vgl. 28, S. 6]

Isolation Forest

Der IF Algorithmus trennt Daten in einem „Wald“ (Isolation Forest) aus sogenannten Isolation Trees, die die Datenpunkte rekursiv aufteilen, bis jeder Datenpunkt bzw. jede Gruppe identischer Datenpunkte isoliert ist. Wichtig ist, dass sich die Anomaliedatenpunkte am Anfang des Isolation Forest befinden. Die Pfadlänge eines zu klassifizierenden

Datenpunkts, also die Anzahl der Kanten, die traversiert wurden, bevor ein Endknoten erreicht wird, bestimmt den Anomalie-Score des Datenpunkts. Der Anomalie-Score entscheidet, ob der Datenpunkt als normal oder abnormal klassifiziert wird. [vgl. 33, S. 287-288]

K-Means

Der KM Algorithmus gehört zu den Clustering-Algorithmen. Das K im Namen steht für die Anzahl an Clustern, die gebildet werden sollen. Der Algorithmus bildet die Anzahl der Cluster-Mittelpunkte also nicht dynamisch, um den Fehler zu minimieren. Stattdessen wird versucht, die Cluster-Mittelpunkte so zu verschieben, dass die Summe der quadratischen Distanzen von den Datenpunkten zu den jeweils nächstgelegenen Cluster-Mittelpunkten minimiert wird. Der Algorithmus beginnt damit, dass die K Cluster-Mittelpunkte an zufällige Orte im Merkmalsraum der Daten gesetzt werden. Der Algorithmus wiederholt nun zwei Phasen, um den Fehler sukzessive zu minimieren, bis er konvergiert. In der ersten Phase werden die Datenpunkte den Clustern zugeordnet, deren Mittelpunkt ihnen am nächsten liegt. In der zweiten Phase wird der Mittelpunkt der Cluster in den Mittelpunkt der dem Cluster zugeordneten Datenpunkte verschoben. [vgl. 3, S. 424–426]

Mean Shift

Der MS Algorithmus gehört zu den Clustering-Algorithmen und bildet Cluster über die Dichteverteilung der Datenpunkte. Die Grundidee besteht darin, alle Datenpunkte sukzessive zu verdichten, damit sich sogenannte Modes bilden, also lokale Dichtezentren. Man verschiebt die Datenpunkte also iterativ in die Richtung des Durchschnitts der Nachbarn, bis sie sich einem Fixpunkt annähern. Die Datenpunkte, die zum gleichen Punkt konvergieren, werden zu einem Cluster zusammengefasst. Im Gegensatz zum K-Means-Algorithmus wird die Anzahl der Cluster also dynamisch bestimmt. [vgl. 4, S. 767–768]

Histogram-based Outlier Score

Mithilfe des Histogram-based Outlier Score (HBOS) können normale Daten von Anomalien getrennt werden. Der Ansatz ist simpel: Für jedes Merkmal bzw. jede Dimension der

Daten im Datensatz wird ein Histogramm erstellt. Dabei werden jeweils Intervalle der Wertebereiche zu gleich großen Behältern zusammengefasst, wobei die Intervallgröße der Behälter statisch angegeben oder dynamisch bestimmt werden kann. Die relative Anzahl der Werte bzw. die Höhe der Behälter repräsentiert dabei eine Dichteabschätzung und bildet die Gewichtung. Damit alle Merkmale gleich gewichtet werden, werden die Behälter innerhalb der Dimensionen so normalisiert, dass der Behälter mit den meisten Daten auf 1.0 normalisiert wird. Mithilfe der normalisierten Histogramme kann dann der HBOS der Dateninstanzen berechnet werden. [vgl. 13, S. 60–62]

Autoencoder

Die Idee des AE ist es, Daten zu kodieren und zu dekodieren. In der Kodierung werden die Dateninstanzen, betrachtet als Merkmalsvektoren, so kodiert, dass sie auf das Wesentliche reduziert werden, wobei auch die Dimensionen der kodierten Instanz reduziert werden. In der Dekodierung geht es darum, die Dateninstanz aus ihrer kodierten Version, und damit auch ihre Dimension, wiederherzustellen. Der Autoencoder wird darauf trainiert, den Rekonstruktionsfehler bei normalen Dateninstanzen, also die Differenz zwischen dem Original und der wiederhergestellten Version, zu minimieren. Die Idee ist, dass der Rekonstruktionsfehler bei normalen Dateninstanzen gering, bei Anomalien aber sehr hoch ist und man damit ein solides Entscheidungskriterium erhält [vgl. 7, S. 60–62].

2.7 Evaluationsmetriken

Um die Ergebnisse eines Anomaliedetektors evaluieren zu können, werden Metriken benötigt, die die Leistung quantifizieren können. Dafür sollte zunächst der Begriff der Confusion Matrix (Konfusionsmatrix) geklärt werden, in der dargestellt werden kann, zwischen welchen Klassen es welche Verwechslungen gegeben hat. Diese Matrix stellt eine Art Tabelle dar, die bei einer Klassifikation in zwei Klassen (Anomalie und Normal) eine Größe von 2 x 2 besitzt. Die Größe stammt daher, dass es zwei Möglichkeiten für die tatsächliche und zwei für die vorhergesagte Klassifikation gibt. Eine solche Matrix findet sich auch in Abbildung 2.5, in der man auf der horizontalen Achse die tatsächlichen Klassen findet und auf der vertikalen Achse die vorhergesagten. Dabei ergeben sich vier Felder: Ein True Positive (TP) kommt vor, wenn eine Dateninstanz korrekterweise

als Anomalie klassifiziert wurde. Ein FP kommt vor, wenn eine Dateninstanz fälschlicherweise als Anomalie klassifiziert wurde, und man könnte es umgangssprachlich als „falschen Alarm“ bezeichnen. Ein False Negative (FN) kommt vor, wenn eine Dateninstanz fälschlicherweise als normal klassifiziert wurde, und bei einem True Negative (TN) wurde eine Dateninstanz korrekterweise als normal klassifiziert. [vgl. 22, S. 181]
Um nun die Vergleichbarkeit sowohl zwischen den Verfahren untereinander als auch zu den Arbeiten anderer herstellen zu können, werden die folgenden Evaluationsmetriken verwendet, die eine Aussage über die Güte eines Klassifikators treffen können und aus den oben genannten Feldern der Confusion Matrix abgeleitet werden können [vgl. 22, 181–183]:

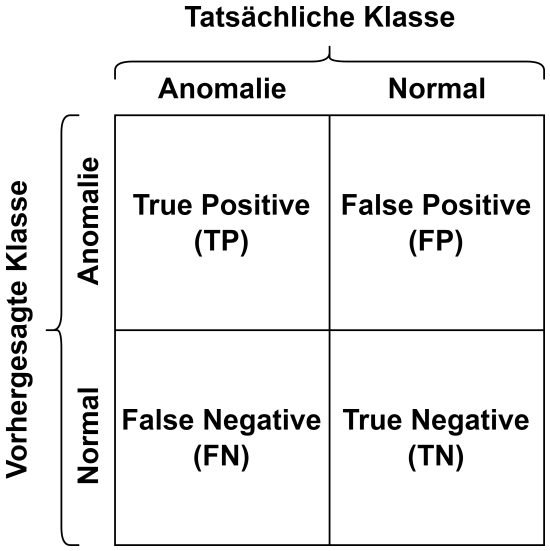


Abbildung 2.5: Die Abbildung zeigt eine Confusion Matrix, die darstellen soll, wie die Klassen Anomalie und Normal miteinander verwechselt werden können.

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP + TN}$$

Die FPR gibt an, wie viele FPs für eine tatsächlich normale Dateninstanz existieren.

$$\text{Precision} = \frac{TP}{TP + FP}$$

Die Präzision gibt den Anteil der tatsächlichen Anomalien unter den als Anomalie klassifizierten Dateninstanzen an.

$$\text{Recall} = \frac{TP}{TP + FN}$$

Der Recall gibt den Anteil der korrekt erkannten unter den vorhandenen Anomalien an.

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Der F1-Score oder F-Score gibt das harmonische Mittel von Präzision und Recall an und stellt somit ein Maß für die Balance zwischen diesen beiden Werten dar.

2.8 Datenfusion

Die **Datenfusion** ist ein Prozess, bei dem Sensordaten oder aus Sensordaten abgeleitete Informationen zu einer gemeinsamen Darstellung zusammengeführt werden. Dabei werden verschiedene theoretische Ansätze, Techniken und Werkzeuge eingesetzt, um die Informationen aus mehreren Quellen zu integrieren. Das Ziel dieser Vorgehensweise ist es, die Qualität der verfügbaren Informationen zu verbessern, sodass diese insgesamt verlässlicher und aussagekräftiger sind, als es bei der Nutzung einzelner Sensoren der Fall wäre.

2.8.1 Grundlegende Fusionsbegriffe

Vor der Frage nach Kategorisierungen für die Funktionsweise verschiedener Fusionsverfahren sollte zunächst ein Blick auf die konkreten Ziele gelenkt werden, die mit der Sensorfusion verbessert werden sollen. Hier gibt es vier Schlüsselziele, die es zu verbessern gilt: [vgl. 21, S. 1–2]:

- **Representation**: Die Informationen, die im Prozess der Fusion gewonnen werden, besitzen ein höheres Abstraktionsniveau oder eine bessere Granularität als die Eingabedaten. Dies bedeutet, dass durch die Datenfusion eine höhere Semantik oder Aussagekraft erreicht wird.

- **Certainty**: Die Datenfusion bringt mehr Sicherheit für die Korrektheit der Daten. Man könnte sagen, dass die Wahrscheinlichkeit für die Korrektheit durch die Datenfusion zunimmt.
- **Accuracy**: Die Datenfusion sorgt dafür, dass die Standardabweichung in den fusionierten Daten kleiner ist als in den originalen, einzelnen Daten. Dies ist vor allem im Hinblick auf Rauschen und Fehler in den Daten wichtig, wenn der Fusionsprozess darauf abzielt, diese zu reduzieren.
- **Completeness**: Die Datenfusion bringt neue Sichtweisen oder Wissen zur Umgebung der Daten. Gibt es Überdeckungen der ursprünglichen und neu hinzugefügten Daten, so kann man auch von einem Zugewinn an Accuracy sprechen.

Neben den verschiedenen Wegen, wie Datenfusion genutzt werden kann, gibt es auch verschiedene Strategien oder Methoden, wie fusioniert werden kann [vgl. 21, S. 3]:

- **Fusion across sensors**: Bei der Fusion zwischen Sensoren werden die Daten verschiedener Sensoren, die dieselbe Größe messen, zusammengeführt und fusioniert.
- **Fusion across attributes**: Bei dieser Fusion werden im gleichen Kontext verschiedene Attribute gemessen und fusioniert. Dies ermöglicht z. B. eine Fusion von Temperatur, Druck und Luftfeuchtigkeit zum Brechungsindex der Luft, womit die Completeness erhöht wird.
- **Fusion across domains**: Bei der Fusion zwischen Domänen werden die Daten verschiedener Sensoren fusioniert, die zwar dasselbe Attribut (z. B. die Temperatur), aber nicht im gleichen Kontext messen. Dieser Kontext kann ein anderer Wertebereich oder eine andere Domäne sein.
- **Fusion across time**: Bei der Fusion über die Zeit können die Daten eines Sensors mit der Historie seiner Daten fusioniert werden.

Weiterhin existieren im Kontext der Sensorfusion verschiedene Konfigurationen für eine Fusion [vgl. 21, S. 4]:

- **Complementary**: Die Konfiguration einer Sensorfusion wird als komplementär bezeichnet, wenn die Sensoren nicht voneinander abhängig sind und ihre Fusion ein größeres Gesamtbild erzeugt. Diese Konfiguration stellt sich also dem Problem der Unvollständigkeit.

- **Competitive**: Die Sensoren einer Fusion stehen sich kompetitiv gegenüber, wenn sie eine unabhängige Messung derselben Größe vornehmen und es darum geht, einen Sieger zu ermitteln, um Ungewissheiten und Fehler zu reduzieren.
- **Cooperative**: Bei der kooperativen Konfiguration einer Sensorfusion ist es das Ziel, neue Informationen aus den einzelnen Sensordaten abzuleiten, die ohne eine Fusion nicht verfügbar gewesen wären.

Zuletzt bestehen verschiedene Möglichkeiten für den Zeitpunkt der Fusion. Hauptsächlich wird in drei Level oder Ebenen unterschieden, also unterschiedliche Zeitpunkte im Verarbeitungsprozess [vgl. 14, S. 16–17]:

- **Data level fusion**: Bei der Fusion auf dem Daten-Level geht es darum, verschiedene Sensorquellen zu einer gemeinsamen Quelle zusammenzuführen. Es werden demnach die Rohdaten fusioniert.
- **Feature level fusion**: Auf dem nächsten Level werden verschiedene Merkmale fusioniert, um den Merkmalsraum auf weniger, aber dafür aussagekräftigere Dimensionen zu reduzieren. Bekannte Verfahren, die auf dieser Ebene eingesetzt werden können, sind die Principal Component Analysis (PCA) oder die Correlation-Based Feature Selection (CFS).
- **Decision level fusion**: Auf der höchsten Ebene werden die Entscheidungen verschiedener Quellen zu einem Ergebnis fusioniert.

2.8.2 Fusionsanwendungen

Datenfusion wird in vielen verschiedenen Feldern eingesetzt. Im Kontext von Smart Home Systemen werden in der Arbeit [34] verschiedene Informationen aus der Umgebung zusammengetragen, um einen geeigneten Zeitpunkt für die Wecker-Zeit zu ermitteln. Hierfür werden zwei Ansätze miteinander verglichen: Ein Fuzzy-Logic-Modul, das mit if-then-Regeln eine geeignete Zeit ermitteln soll, steht einem Fuzzy Neural Network gegenüber, welches im direkten Vergleich bessere Ergebnisse erzielt.

Im Kontext der Zielerkennung beschäftigt sich die Arbeit von Xiao et al. [32] mit dem Einsatz der D-S Evidenztheorie für die Datenfusion. Hierbei wurden die lokalen Entscheidungen der drei verwendeten Erkennungssysteme, die unterschiedliche bildgebende

Sensoren verwenden, fusioniert. So konnte das fusionierte System mit zuverlässigeren und stabileren Ergebnissen im Vergleich zu den einzelnen Systemen überzeugen.

In der Arbeit von Fessi et al. [12] wurde Datenfusion für ein IDS eingesetzt. Im Netzwerk verteilt finden sich einige Beobachter, deren Aufgabe es ist, Daten zu sammeln und an ein zentrales System zu senden. Innerhalb des zentralen Systems existieren mehrere Analyse-Systeme, die jeweils unterschiedliche Techniken für die Intrusion Detection verwenden. Die Datenfusion erfolgt dabei auf dem Decision-Level und fusioniert die Teilergebnisse der Analyse-Systeme zu einem Gesamtergebnis. Insgesamt ermöglicht dieser Ansatz eine präzisere und robustere Erkennung von Angriffen, indem die Stärken der unterschiedlichen Analyse-Techniken effizient kombiniert werden.

3 Zielsetzung

Das übergeordnete Ziel dieser Arbeit ist zweigeteilt. Das erste Ziel umfasst die Bündelung und Verknüpfung bereits bestehender Werkzeuge (Vorgefertigte Simulationsumgebungen, Bibliotheken für die Analyse von Netzwerkaufzeichnungen oder für ML etc.) in ein Framework, welches möglichst effizient und automatisiert dazu in der Lage sein soll, Anomalien in Netzwerkaufzeichnungen von IVNs zu erkennen. Um eine effiziente Anomalieerkennung vornehmen zu können, sollen verschiedene Maßnahmen beleuchtet werden, die es bspw. ermöglichen, nicht notwendige Operationen einzusparen und die vorhandenen Rechenkapazitäten durch Parallelität auszunutzen. Es umfasst außerdem die Erstellung besagter Aufzeichnungen in einer simulierten Umgebung, in der verschiedene Arten von Anomalien hervorgerufen werden können. Es sollen verschiedene Anomalieerkennungsalgorithmen eingesetzt und die Klassifikationsergebnisse evaluiert werden können, sodass ein möglichst direkter Vergleich zwischen verschiedenen Algorithmen und ihren Konfigurationen möglich ist.

Darauf aufbauend umfasst das zweite Ziel die Fusion von verschiedenen Anomalieerkennungssystemen. Aufgabe der Datenfusion ist die Verbesserung der Erkennungsleistung und die Reduzierung von Fehlalarmen. Auch für das zweite Ziel gilt: Es sollen vielversprechende Verfahren implementiert werden, sodass ein möglichst hoher Erkenntnisgewinn verzeichnet werden kann. Ausdrücklich nicht zum Ziel gehört die Umsetzung der Ergebnisse in ein funktionsfähiges und einsatzbereites, verteiltes Anomalieerkennungssystem, welches in realen Fahrzeugen eingesetzt werden kann. Dennoch darf im Prozess nichts unternommen werden, was dazu führt, dass die Ergebnisse nicht umsetzbar sind. So darf bspw. kein Vorwissen genutzt werden, das in einer realen Umsetzung nicht zur Verfügung stehen würde. Durch die Datenfusion soll der Anomaliedetektor in der Lage dazu sein, der alle Anomalien erkennen kann, ohne dabei Fehlalarme auszulösen.

Neben den beiden Hauptzielen stellt insbesondere der Umgang mit Fehlalarmen eine zentrale Herausforderung bei der Anomalieerkennung dar. Da die Bedeutung eines Fehlalarms (konkret FP) stark vom jeweiligen Kontext abhängt, soll im Rahmen dieser Arbeit

der Kontext ihres Auftretens systematisch analysiert und in die Evaluation miteinbezogen werden.

Zusammenfassend ist es Ziel dieser Arbeit, dass zunächst ein Framework für die Erstellung von Testdaten entstehen soll, welches verwendet werden soll, um verschiedene Fusionsverfahren für den Einsatz in der Anomalieerkennung zu testen und ihre Ergebnisse zu vergleichen. In die Evaluation soll außerdem das Auftreten von FPs einbezogen werden, um eine möglichst differenzierte Betrachtung der Ergebnisse zu ermöglichen.

In den folgenden Sektionen sollen zunächst die allgemeinen Evaluationsmetriken eingeführt werden. Weiterhin soll eine Motivation darüber gegeben werden, warum die Metriken als besonders aussagekräftig angesehen werden. Anschließend sollen jeweils getrennt die Teilziele für Anomalieerkennung und Datenfusion vorgestellt und genauer spezifiziert werden.

3.1 Evaluationsmetriken

Die Evaluationsmetriken für Anomalieerkennung und Datenfusion bauen direkt auf den Feldern der Konfusionsmatrix auf, die bereits in den *Grundlagen* (Kapitel 2) vorgestellt wurden. Allgemein ist das Ziel, die absolute Zahl der Fehlklassifikationen auf 0 zu bringen. Da dies in der Realität vermutlich nicht umsetzbar sein wird, sollte ein Wert zur Verfügung stehen, der das relative Verhältnis der Fehlklassifikationen zum Rest angeben kann. Der *Recall* gibt an, wie viele Anomalien unter den tatsächlich vorhandenen gefunden wurden, und ist damit das wichtigste Maß für die Erkennungsleistung eines Anomaliedetektors. Sind alle Anomalien detektiert worden, beträgt der Wert 1. Der Recall allein besitzt das Problem, dass die FPs nicht in die Rechnung einfließen und ein perfekter Wert auch erzielt werden kann, indem alles als Anomalie klassifiziert wird. Aus diesem Grund wird auch die *Precision* ausgerechnet, die ein Maß dafür ist, wie viele Dateninstanzen unter allen klassifizierten Anomalien auch wirklich Anomalien sind. Um die beiden Werte ins Verhältnis zueinander setzen zu können, kann der *F1-Score* berechnet werden, der das harmonische Mittel der beiden Werte darstellt. Für das Verhältnis der FPs im Gesamtergebnis reicht die Precision nicht aus, da sie immer im Verhältnis zur tatsächlichen Anzahl der Anomalien steht. Ist der Anteil der Anomalien in den Daten sehr gering, ist die Precision schnell sehr niedrig, da so auch sehr wenige FPs ausreichen, um die Precision stark zu verschlechtern: Gibt es im Ergebnis insgesamt 1.000.000

Datenpunkte, von denen 10 Anomalien sind und auch als solche erkannt wurden, reichen 10 FPs, um die Precision auf $\frac{10}{10+10} = 0,5$ zu setzen, was ein relativ schlechter Wert ist. Aus einem anderen Winkel betrachtet, könnte man die FPs auch ins Verhältnis zu den normalen Datenpunkten setzen. So ergibt sich die sogenannte *FPR*, die in dem Beispiel bei $\frac{10}{10+999.980} = 0,00001$ liegt. Dieser Wert spiegelt besser wider, wie viele Dateninstanzen korrekterweise als normal klassifiziert wurden. In der Realität könnte eine Gegenmaßnahme für erkannte Anomalien in Form einer Warnung an den Fahrer erfolgen. Um einschätzen zu können, wie oft der Fahrer mit falschen Meldungen konfrontiert werden würde, kann man die Anzahl der FPs außerdem ins Verhältnis zur Fahrtzeit setzen. So sollen neben den bereits vorgestellten Metriken auch die *FPs pro gefahrener Stunde* ausgerechnet werden.

3.2 Anomalieerkennung

Um das erste Teilziel umsetzen zu können, muss zunächst geklärt werden, welche Werkzeuge für die einzelnen Schritte genutzt werden können. Anschließend muss beleuchtet werden, wie die Werkzeuge kombiniert und verknüpft werden können. Das für die Datenbeschaffung verwendete Simulationstool, dessen Funktion und Einsatz in den Kapiteln *Konzept* (Kapitel 4) und *Umsetzung* (Kapitel 5) genauer erklärt werden, ist in der Lage dazu, Netzwerkaufzeichnungen von verschiedenen Anomalieszenarien in einem vordefinierten Fahrzeugnetzwerk zu erstellen. Dabei kann die Häufigkeit für das Auftreten einer Anomalie über einen Wahrscheinlichkeitsparameter gesteuert und ein Ablauf angegeben werden, sodass sich die Häufigkeiten von Anomalien im Laufe der Zeit verändern können. Außerdem werden alle erzeugten Pakete mit einem Label versehen, welches angibt, ob das Paket manipuliert wurde, was die Evaluation der Ergebnisse ermöglicht. In diesem ersten Schritt gibt es demnach zwei Stellschrauben, an denen Änderungen vorgenommen werden können, um unterschiedliche Szenarien zu erstellen: das Anomalieszenario und der Ablauf des Szenarios. Konkret sollen die Szenarien die Elimination, Injection sowie das Reordering und Delaying von Frames umfassen.

Auf der nächsten Ebene existieren verschiedene Netzwerkmetriken und Beobachtungsschwerpunkte, die als Grundlage der Anomalieerkennung dienen können. Für die Anomalieerkennung sollen die Algorithmen EE, SVM, AE, KM, MS, HBO und IF eingesetzt werden. Dabei gibt es verschiedene Parametrisierungsoptionen, mit denen die Algorithmen konfiguriert werden können. Der Einfluss der Konfigurationen sowie die generelle

Eignung der Algorithmen sollen durch das Framework systematisch evaluiert werden können.

Zusammenfassend soll ein Framework auf Basis bestehender Werkzeuge entwickelt werden, mit dem es möglich ist, die Konfigurationen verschiedener Beobachtungsschwerpunkte und Algorithmen in Bezug auf verschiedene, definierbare Anomalieszenarien testen und evaluieren zu können. Dafür sollen die Precision, der Recall, der F1-Score, die FPR und die FPs pro Stunde zusammen mit den tatsächlichen und vorhergesagten Klassifikationen ausgegeben werden können. Dabei ist insbesondere wichtig, dass alle möglichen Kombinationen der verschiedenen Konfigurationen automatisch bestimmt und die Anomalieerkennung ausgeführt werden.

3.3 Datenfusion

Der Weg zur Datenfusion ist geradliniger als der zur Anomalieerkennung. So ist es für Fusion lediglich wichtig, geeignete Verfahren zu bestimmen, sie zu implementieren und anhand geeigneter Metriken zu evaluieren. Für die Entscheidung darüber, welche Verfahren ausgewählt werden, ist es von großer Bedeutung, auf welcher Ebene fusioniert werden soll. Dafür gibt es (wie bereits in den *Grundlagen* (Kapitel 2) erklärt) drei Möglichkeiten. Die Fusion auf der höchsten Ebene, dem sogenannten Decision Level, ist sehr interessant, da jeder Anomaliedetektor Fehler machen kann. Aufgrund dessen ist das übergeordnete Ziel der Fusion klar: die Anzahl von Fehlern reduzieren und somit die Certainty durch den Fusionsprozess erhöhen. Die beiden unteren Ebenen (Data und Feature Level Fusion) fallen in der Arbeit aus folgenden Gründen aus: Auf dem Data Level werden die Daten verschiedener Sensoren fusioniert. Da es sich in dieser Arbeit speziell um Netzwerke handelt, sind die Möglichkeiten der Sensorik beschränkt, sodass nur detektiert oder aufgezeichnet werden kann, wann und wo ein Paket oder ein Frame übertragen wurde. Anders als bspw. bei bildgebenden Verfahren existieren auch weder verschiedene Sensortechnologien noch Messfehler bei der Aufzeichnung, was die Ziele der Fusion begrenzt. Die Certainty ist immer nahezu perfekt, da es keine Fehler bei der Aufzeichnung selbst in Form von unbemerkten Paketen o.Ä. geben kann und auch die Accuracy stets perfekt ist, weil es kein Rauschen auf der logischen Übertragung gibt. Auch bieten die 'Sensoren' im Netzwerk stets ein komplettes Bild, sodass die Fusion auch für die Completeness irrelevant ist. Auch das letzte Ziel, die Representation, kann

mit einer Fusion auf dem Data Level kaum erhöht werden, da die Übertragung lediglich an verschiedenen Stellen aufgezeichnet werden kann. Ein höheres Abstraktionslevel könnte zwar erlangt werden, indem bspw. ein Überblick über die Pakete erstellt wird und so der Verlust von Paketen verfolgt werden könnte. Doch würde dies in der realen Welt einerseits sehr aufwendig sein, da eine Art zentrale Datenbank mit potenziell Millionen Einträgen aktuell gehalten werden müsste, und der Verlust einzelner Pakete andererseits kaum als Anomalie gewertet werden kann, da dies kaum vermeidbar ist und darüber hinaus Technologien existieren, die mit solchen Verlusten umgehen können. In dem Fahrzeugnetzwerk, welches in dieser Arbeit verwendet und im *Konzept* (Kapitel 4) genauer erklärt wird, existiert bspw. eine natürliche Redundanz, da Daten auf verschiedenen Wegen parallel übertragen und später zusammengeführt werden können. Somit ist der Verlust einzelner Pakete weder ein großes Problem noch interessant für die Einleitung von Gegenmaßnahmen, da diese potenziell in sehr hoher Frequenz und ohne zugrundeliegenden Angriff ausgeführt werden müssten.

Die Datenfusion auf der zweiten Fusionsebene (Feature Level) ist im Kontext der Anomalieerkennung in Netzwerken bereits interessanter, da aus den Rohdaten im Netzwerk verschiedene Merkmale extrahiert werden können und mittels Methoden wie der PCA[1] fusioniert werden könnten, um weniger, aber aussagekräftigere Merkmale zu erhalten. Eine Umsetzung auf dieser Ebene würde jedoch die Entwicklung eines eigenen Frameworks erfordern, was den Rahmen dieser Arbeit deutlich sprengen würde und daher besser als eigenständiges Thema für eine Facharbeit geeignet wäre. Zusammenfassend ist das Ziel der Datenfusion, die Anzahl von Fehlklassifikationen zu minimieren. Dafür sollen Verfahren ausgewählt und implementiert werden, die auf dem Decision Level die Ergebnisse verschiedener Anomalieerkennungssysteme fusionieren. In Bezug auf die genutzten Metriken bedeutet dies, dass alle Anomalien gefunden werden, dass keine Falsch-Positiven Ergebnisse auftreten und somit Precision, Recall und F1-Score bei 1 liegen, während die FPR und die FPs pro Stunde bei 0 liegen.

[1] PCA steht für *Principal Component Analysis* (Hauptkomponentenanalyse). Dabei handelt es sich um ein statistisches Verfahren zur Reduktion der Dimensionalität von Datensätzen. Es wird häufig eingesetzt, um komplexe Daten übersichtlicher darzustellen oder für die weitere Verarbeitung vorzubereiten.

4 Konzept der Anomalieerkennung

In diesem Kapitel soll erläutert werden, wie der konzeptionelle Aufbau der Datenverarbeitung aussieht. In jeder Sektion findet sich zunächst eine Problem- bzw. Zielstellung. Anschließend werden mögliche Lösungen diskutiert, von denen im letzten Schritt eine ausgewählt und näher vorgestellt wird. Das übergeordnete Ziel, für welches das Konzept entworfen wurde, ist es, eine möglichst gute Anomalieerkennung in Fahrzeugnetzwerken zu entwickeln.

Die erste Sektion beschäftigt sich mit dem Fahrzeugnetzwerk, in dem Anomalien erkannt werden sollen. Die zweite Sektion behandelt die Frage, welche Beobachtungsschwerpunkte im Netzwerk gesetzt werden sollten. Die dritte Sektion diskutiert die verschiedenen Möglichkeiten der Anomalieerkennung. In der letzten Sektion wird sich mit der Frage auseinandergesetzt, wie Datenfusion genutzt werden kann, um die Ergebnisse der Anomalieerkennung zu verbessern.

4.1 Das Fahrzeugnetzwerk

In dieser Sektion des Konzepts wird das Fahrzeugnetzwerk beschrieben, das verwendet wird, um die Anomalie- und Trainingsdaten zu generieren. Auch der Prozess der Generierung soll nähergebracht werden. Grundsätzlich gibt es zwei mögliche Quellen für die Daten: eine Simulationsumgebung oder einen Aufbau in tatsächlicher Hardware. Dabei gibt es für beide Optionen Pro- und Kontra-Argumente. Ein typisches Problem von Simulationen ist die Realitätsnähe. Es stellt sich also die Frage, ob die Simulationsumgebung präzise genug in der Lage dazu ist, die Realität abzubilden. Da es sich in dieser Arbeit um ein rein digitales System handelt, ist eine Simulation grundsätzlich leichter umzusetzen. Es müssen weder physikalische Effekte wie Materialalterung noch Umwelteinflüsse oder Signalverzerrungen berücksichtigt werden.

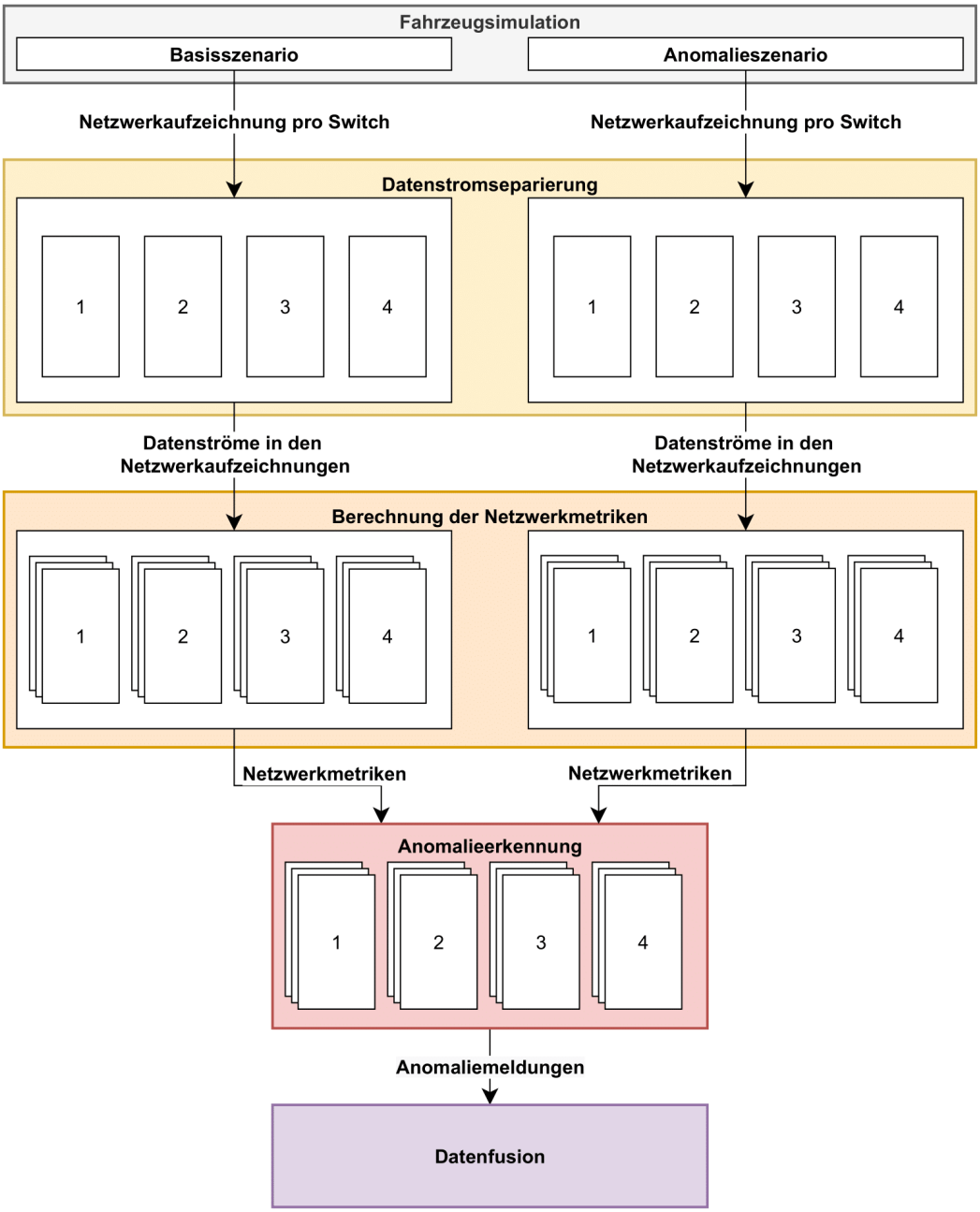


Abbildung 4.1: Die Abbildung visualisiert das Konzept der Datenverarbeitung, das aus den fünf Schritten Fahrzeugsimulation, Datenstromseparierung, Berechnung der Netzwerkmetriken, Anomalieerkennung und Datenfusion besteht.

Ein weiterer Punkt betrifft die Ausführungszeit der Simulation. Je nachdem, wie effizient die Simulation arbeitet und wie viel simuliert werden muss, kann die Simulationsgeschwindigkeit ein Vor- oder Nachteil gegenüber einem Hardwareaufbau sein.

Ein klarer Vorteil der Simulationsumgebung ist die Vielfältigkeit der Szenarien. Da volle Kontrolle über die Simulation besteht, können Szenarien ausgeführt werden, die in Hardware nicht oder nur schwer abzubilden sind, was gerade in sicherheitskritischen Domänen relevant ist. Auch für Parameterstudien, in denen sehr viele Simulationen durchgeführt werden müssen.

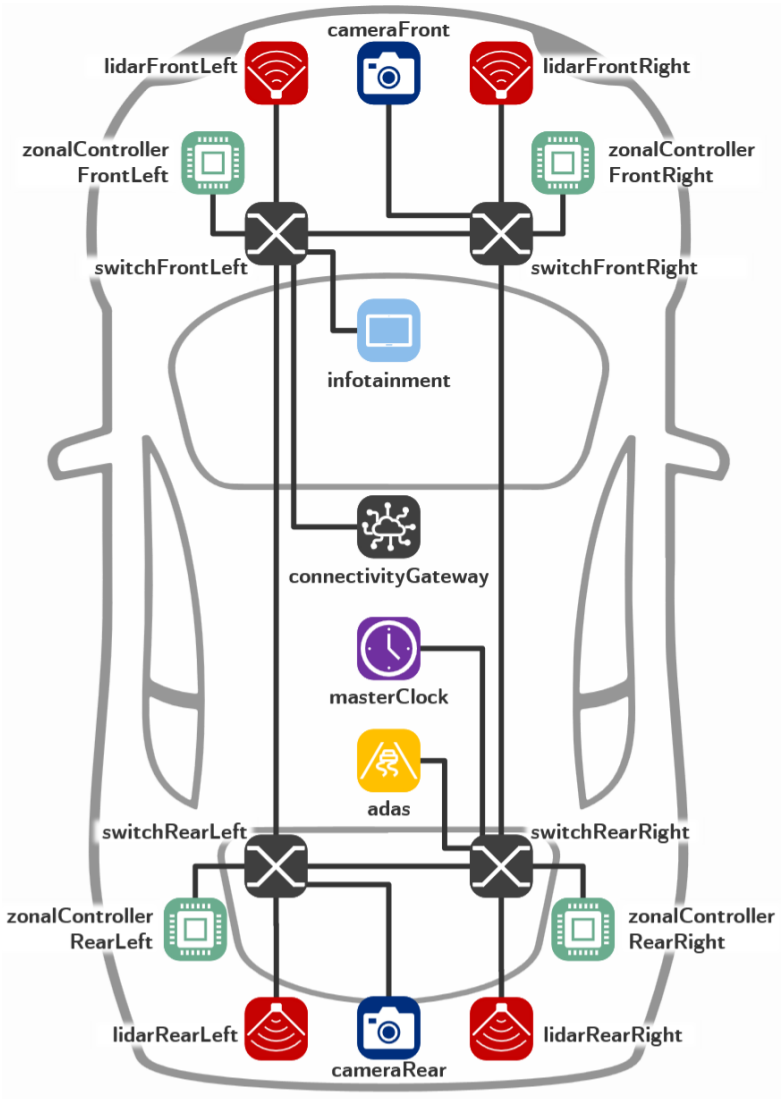
Simulationsumgebung

Die in dieser Arbeit verwendete Simulationsumgebung ist OMNeT++, eine modulare, C++-basierte Simulationsplattform, die primär für die Konstruktion und Simulation von Netzwerken entwickelt wurde. OMNeT++ selbst ist dabei kein Netzwerksimulator, stellt aber die notwendigen Werkzeuge wie bspw. den Simulationskernel, die Network Description Language, mit der Netzwerktopologien definiert werden können, und eine IDE bereit. [9]

Das INET-Framework ist eine Open-Source-Bibliothek für die OMNeT++-Simulationsplattform und wird vom OMNeT++-Team und der Community weiterentwickelt. Es werden die Protokolle des Internet-Stacks (TCP, UDP, IPv4, IPv6, Ethernet etc.), Agenten (Switches, Router etc.) und andere Modelle bereitgestellt, mit denen Kommunikationsnetzwerke simuliert werden können. Die Architektur des INET-Frameworks baut dabei auf Modulen/Komponenten auf, die untereinander via Message-Passing kommunizieren. [8]

Aufbau des simulierten Fahrzeugnetzwerks

Das verwendete Fahrzeugnetzwerk baut auf einem gemeinsamen TSN-Backbone auf und wird in Abbildung 4.2 dargestellt. Die simulierten Controller-Area-Network- CAN-Komponenten stammen aus einem echten Fahrzeug. Die Topologie des Netzwerks besitzt vier redundante Zonal-Controller, die an jeweils einen der vier zentralen Switches angeschlossen sind. Die insgesamt vier Lidar-Sensoren und zwei Kameras, die sich gleichmäßig auf die Vorder- und Rückseite aufteilen, senden ihre Daten über die Switches an das zentrale Advanced Driver Assistance System (ADAS). Auch eine zentrale Uhr wird



Quelle: https://doi.org/10.1109/VNC61989.2024.10576017 [19]

Abbildung 4.2: Die Abbildung zeigt das Fahrzeugnetzwerk, das in OMNeT++ abgebildet wurde und verwendet wird, um die Trainings- und Testdaten zu erzeugen.

mit Zeitdrift simuliert, wobei alle Komponenten über das gPTP synchronisiert werden. Mithilfe des Netzwerkes können sowohl Anomalie- als auch Trainingsdatensätze erstellt werden. [vgl. 19, S. 61] Aufgrund der Größe der erstellten Netzwerkaufzeichnungen ist die Simulationslänge in dieser Arbeit auf zwei Minuten begrenzt. Mit dieser Simulati-

onslänge wird sichergestellt, dass der Rechenaufwand in einem hinnehmbaren Rahmen bleibt, und dennoch auch die langsamsten CAN-Intervalle (siehe Abbildung 4.3) mindestens 60 Mal in der Aufzeichnung zu finden sind. Die Größe der erzeugten Dateien für ein Szenario liegt dabei schon bei ca. 40 GB.

Datenströme im simulierten Fahrzeugnetzwerk

Im Fahrzeugnetzwerk gibt es zahlreiche Datenströme, die in Tabelle 4.1 aufgeführt sind. Für die Anomalieerkennung sind insgesamt 216 dieser Ströme von Bedeutung. Der Großteil entfällt dabei auf die sogenannten Control-Source-Apps – auch wenn ihr Anteil am gesamten Datenverkehr eher gering ist. Die bandbreitenintensivsten Ströme stammen hingegen von den Kamera- und LIDAR-Systemen. Zentrale Punkte im Netzwerk sind die vier Zonal-Controller und das Advanced Driver Assistance System (ADAS).

Traffic	**Source**	**Destination**	**Redundancy**
gPTP Sync	(masterClock)	–	2 gPTP-Domains
Manual Throttle / Brake / Steer	Zonal-Controller Front-Left	Zonal-Controller*	802.1CB im Backbone [16]
Automatic Throttle / Brake / Steer	ADAS	Zonal-Controller*	802.1CB im Backbone [16]
Video (2 Source Apps)	camera *	ADAS	802.1CB im Backbone [16]
LIDAR (4 Source Apps)	LIDAR *	ADAS	802.1CB im Backbone [16]
Control (201 Source Apps)	Zonal-Controller*/Infotainment	Zonal-Controller*/Infotainment	–
V2X	Connectivity-Gateway/ADAS	Connectivity-Gateway/ADAS	–
Background			-

Tabelle 4.1: In der Tabelle sind die Datenströme im Fahrzeugnetzwerk mit Start, Ziel und Redundanzen aufgelistet. Das Sternchen(*) steht stellvertretend für alle Instanzen der konkreten Komponente [20].

4.2 Szenarien

Die Konfiguration der verschiedenen Szenarien erfolgt an drei Stellen. Zunächst muss für jede Art von Anomalieszenario (Elimination, Injection …) definiert werden, was geschehen soll. Diese Definitionen finden sich in der Implementierung eines speziell angepassten Link-Layers und werden in dieser Arbeit nicht weiter behandelt, da sie mit zum Framework gehören. An dieser Stelle könnten jedoch mehr Arten von Anomalien hinzugefügt werden. Weiterhin nicht behandelt werden die Definitionen des Netzwerks, in dem der korrupte Link-Layer verbaut ist. Während konfiguriert ist, dass keine Anomalien auftreten sollen, sind diese Fahrzeugnetzwerke semantisch identisch mit dem originalen Netzwerk, das für die Erzeugung der Trainingsdaten verwendet wird. Die Konfiguration der entsprechenden korrupten Link-Layer-Instanz erfolgt über XML-Dateien. In diesen Dateien können bspw. der betroffene Datenstrom, die Wahrscheinlichkeit für das Auftreten einer Anomalie und ein Mindestabstand zwischen zwei Anomalien konfiguriert werden.

Da nun geklärt ist, wie die Szenarien konfiguriert werden können, stellt sich die Frage, wie die Konfigurationen im Detail aussehen sollen. Dabei gibt es zwei besonders entscheidende Punkte: den Ablauf und die Länge der Szenarien.

Der Ablauf der Szenarien ergibt sich aus der Überlegung, welche Daten erzeugt werden sollten, um möglichst nahe an der Realität zu sein. Dazu kann gesagt werden, dass es zwei mögliche Zustände geben kann, in denen sich ein Fahrzeug im Sinne der Anomalieerkennung befinden kann: in einem Anomalie-Zustand oder nicht. Daher ergeben sich die zwei Zustände und die beiden Übergänge, die in der Simulation enthalten sein sollten: der Übergang vom normalen Zustand zum Anomalie-Zustand und andersrum. Diese Übergänge sind besonders wichtig, wenn man bedenkt, dass eine Fusionsmöglichkeit darin besteht, die Daten über die Zeit zu fusionieren, mit dem Hintergedanken, dass ein einzelnes Anomalie-Intervall vermutlich nicht von einem Angriff oder einer Fehlfunktion stammt. Da ein Fahrzeug in der Realität ab Werk vermutlich eher in einem normalen Zustand starten wird und um dem Fahrzeugnetzwerk Zeit zu geben, die normalen Vorgänge zu starten, beginnt der Ablauf der Simulation ohne Anomalien. Um ein besonderes Augenmerk darauf legen zu können, dass Anomalien erkannt werden, sobald sie auftreten, kann der Übergang vom normalen zum Anomalie-Zustand zwei Mal abgebildet werden. Daher ergibt sich die Abfolge: Normal Anomalie Normal Anomalie.

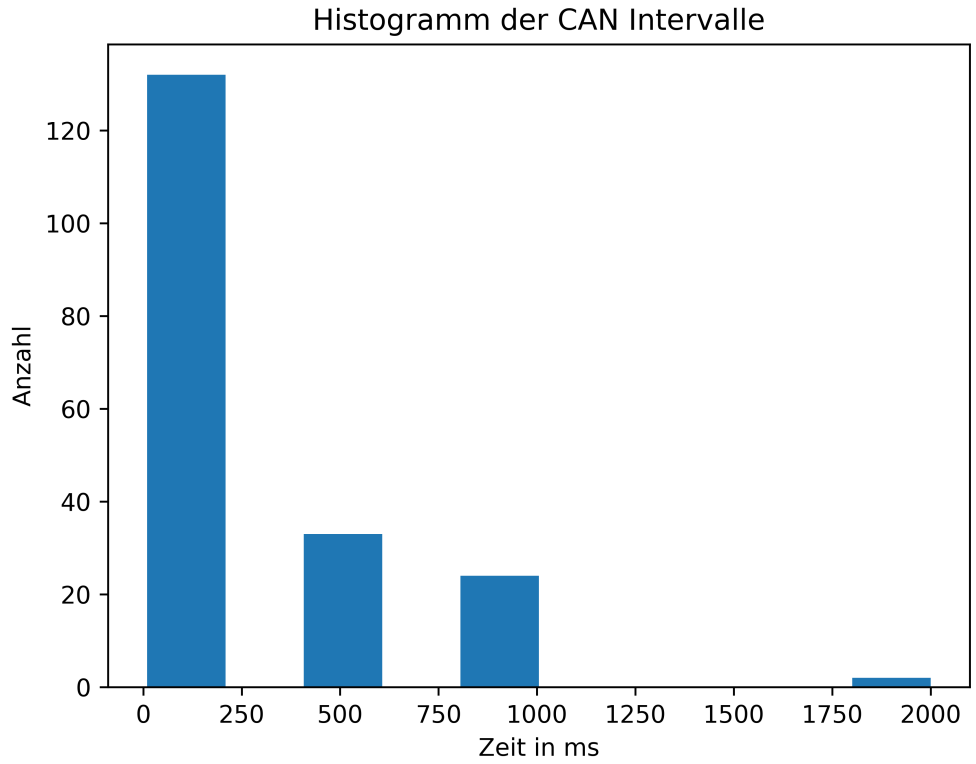


Abbildung 4.3: Die Abbildung zeigt das Histogramm der Timings in den CAN-Datenströmen im simulierten Fahrzeugnetzwerk.

Aufgrund der Datenmenge, die bei der Simulation entsteht, muss eine Abwägung zwischen der Laufzeit der Simulation mit Auswertung und der Vollständigkeit der Abdeckung der möglichen Zustände, in denen sich das Fahrzeugnetzwerk befinden kann, geschehen. Um diese Abdeckung quantifizieren zu können, gibt es mehrere Ansätze. Eine Überlegung betrifft die in Abbildung 4.3 zu sehenden Intervalle der CAN-Komponenten und ihren gesendeten Nachrichten, die beim längsten Intervall alle 2000 ms auftreten. Diese sollten individuell gesehen mindestens einmal vollständig durchlaufen und aufgezeichnet werden. Da alle Pakete, die über eine Netzwerkverbindung (ein Kabel) im Fahrzeugnetzwerk übertragen werden, nicht parallel gesendet werden können und damit in irgendeiner Weise serialisiert werden müssen, haben alle Pakete, die gesendet werden, einen Einfluss auf alle anderen Pakete, die sich zur gleichen Zeit im Netzwerk befinden. Diese unbeabsichtigten und fast unvorhersehbaren Wechselwirkungen sorgen dafür, dass eine vollständige Abdeckung der möglichen Netzwerkzustände nicht erreicht werden

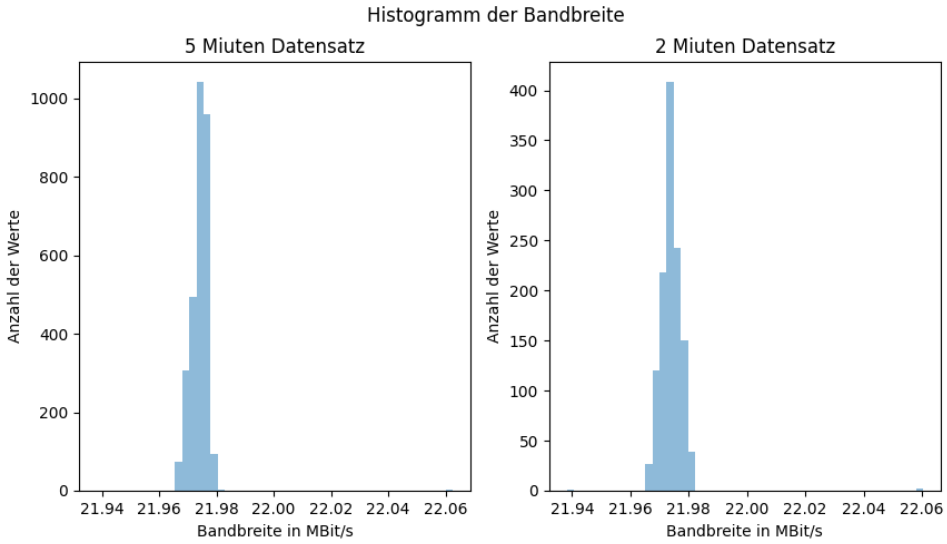


Abbildung 4.4: Die Abbildung zeigt das Histogramm der Bandbreite für 100ms langen Intervallen im Vergleich zwischen einer 2- und einer 5-minütigen Aufzeichnung des LIDAR-Datenstroms im Switch vorne-rechts.

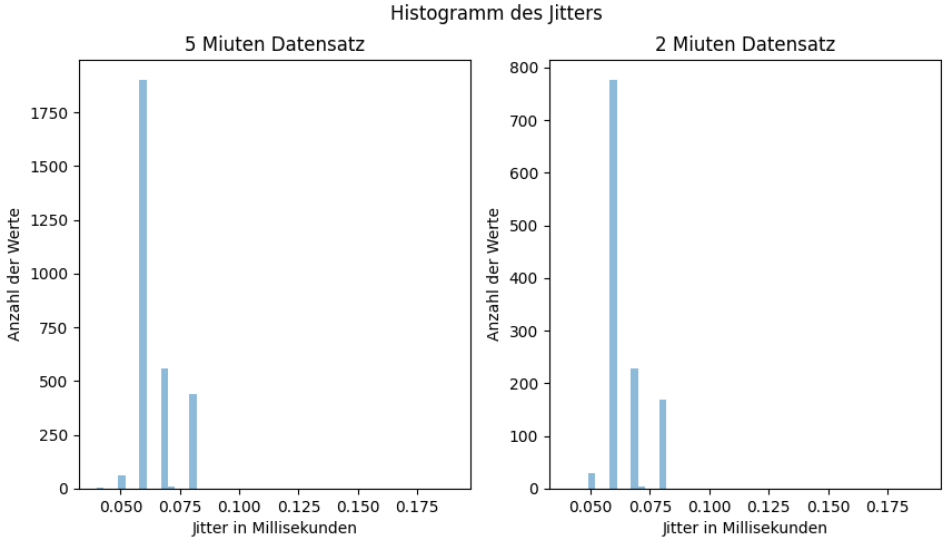


Abbildung 4.5: Die Abbildung zeigt das Histogramm des Jitters für 100ms langen Intervallen im Vergleich zwischen einer 2- und einer 5-minütigen Aufzeichnung des LIDAR-Datenstroms im Switch vorne-rechts.

kann, indem sich die Simulationszeit nach dem langsamsten Kommunikationszyklus von 2000 ms richtet, auch wenn dies ein erster Anhaltspunkt ist. Daher sollten die einzelnen

Teilschritte des Ablaufs zwar mindestens 2000 ms lang sein, aber so lang wie möglich. Diese Überlegung diktiert bei dem zuvor vorgestellten Ablauf mit vier Teilschritten eine Simulationszeit von mindestens acht Sekunden.

Eine weitere Überlegung, die Abdeckung der Zustände des Fahrzeugnetzwerks quantifizieren zu können, liegt in der Beobachtung der berechneten Netzwerkmetriken. In den beiden Abbildungen 4.4 und 4.5 lassen sich die Histogramme der Bandbreite und des Jitters erkennen, die jeweils für Intervalle von 100 ms berechnet wurden. Diese Abbildungen zeigen die Histogramme jeweils im direkten Vergleich zwischen einer fünf- und einer zweiminütigen Netzwerkaufzeichnung des LIDAR-Datenstroms im Switch vorne-rechts im Fahrzeug, welcher auch von einem in dieser Arbeit durchgeführten Anomalieszenario betroffen ist. Der Vergleich soll stellvertretend für das Fahrzeugnetzwerk zeigen, dass sich das Profil der Netzwerkmetriken, die Grundlage für die Bestimmung von Anomalien sind, nicht groß unterscheidet. Da sich die Profile, wie den Abbildungen zu entnehmen, kaum unterscheiden, ist davon auszugehen, dass sich der Informationsgewinn der zusätzlichen 3 Minuten Aufzeichnung in Grenzen hält. Die Laufzeit von Simulation und Datenauswertung bei 2-minütigen Szenarien bewegt sich mit rund 3 Tagen bereits am oberen Ende eines vertretbaren Zeitrahmens, welcher durch die 5-minütigen Aufzeichnungen überschritten wird. Im Vergleich zu der Masterarbeit von Wilhelm Schumacher [23], dessen Trainingsszenario eine für die Simulationsumgebung utopische Laufzeit von 10.000 Sekunden betragen hat und an einem Hardwareaufbau durchgeführt wurde, ist die Laufzeit der Szenarien durch die Auswertungsgeschwindigkeit des Frameworks begrenzt.

Mit den beiden Überlegungen zur Quantifizierung der Abdeckung im Hinterkopf lässt sich die Länge der Simulationen auf 2 Minuten festlegen, da dieser Zeitraum weitaus größer ist als die Mindestanforderung von 8 Sekunden und gleichzeitig einen ähnlichen Informationsgewinn wie längere Zeiträume bietet.

Ausgabedaten

Die Ausgabe der Simulation erfolgt in Form von sogenannten PCAP Next Generation Dump File Format (PCAPNG)-Dateien, mit deren Hilfe der Datenverkehr eines Netzwerks aufgezeichnet und gespeichert werden kann. Dabei liegen die Daten der Pakete oder Frames direkt in der Datei und müssen von einem Parser gelesen und interpretiert

werden. Da es Datenströme im Netzwerk gibt, deren Pakete auf mehreren Routen versendet und später zusammengeführt werden, muss ein potentieller Parser dazu in der Lage sein, mit dem entsprechenden Ethernet-Standard[1] zurechtzukommen. Außerdem findet sich zu jedem Paket ein Paket-Kommentar, der angibt, ob das jeweilige Paket gültig ist oder nicht. Im Falle der Elimination von Paketen gibt es natürlich keine Pakete, daher umfasst der Kommentar des nächsten Pakets nach einer Elimination zusätzlich die Information, dass es das erste Paket nach einer Anomalie ist. Dadurch wird sichergestellt, dass die Anomalieerkennung evaluiert werden kann. Die Anforderungen an den verwendeten Parser geben also vor, dass Paketkommentare und spezielle Ethernet-Erweiterungen interpretiert werden können.

4.3 Beobachtungsschwerpunkte im Netzwerk

Im Bereich der Netzwerkanalyse stellt sich u. A. eine grundlegende Frage: Betrachtet man einzelne Datenströme oder den Netzwerkverkehr als Ganzes? Wenn Angriffe oder ungewollte Verhaltensweisen innerhalb eines Datenstroms auftreten, ergibt es Sinn, die Datenströme getrennt zu betrachten. Gerade bei z. B. Flooding-Attacken ist es hingegen sinnvoll, das Netzwerk als Ganzes zu betrachten, da diese nicht auf einem Datenstrom passieren und nicht erkannt werden können, wenn man alle Datenströme getrennt analysiert. [vgl. 25, S. 551–552]

Entscheidet man sich für die Separierung der Datenströme, kann die Überlegung angestellt werden, ob auf Paket- oder Flow-Ebene analysiert wird. Für die Entscheidung gilt es zu beachten, welche Kommunikation stattfindet: Werden kleinere Nachrichten versendet, die in einem Paket übermittelt werden können, oder größere Nachrichten, die sich über mehrere Pakete erstrecken? Eine weitere relevante Betrachtung ist die Analyse von Intervallen, also der Zusammenhang mehrerer aufeinanderfolgender Pakete, was auch für die Analyse des Netzwerkes als Ganzes möglich ist. Durch die Analyse solcher Intervalle kann ein besseres Verständnis des zeitlichen Verhaltens der Kommunikation gewonnen werden, insbesondere wenn Nachrichten über längere Zeiträume verteilt übertragen werden.

[1] Der Standard *IEEE 802.1CB* beschreibt Mechanismen zur Erhöhung der Netzwerkausfallsicherheit durch **Frame Replication and Elimination**, indem Pakete über mehrere redundante Pfade gesendet und die Duplikate später zusammengeführt werden [16].

Insbesondere bei der Erkennung von Angriffen kann es entscheidend sein, ob einzelne Pakete oder auch Paket-Intervalle untersucht werden. Manche Angriffe basieren auf Anomalien innerhalb einzelner Pakete, während andere, komplexere Angriffe über mehrere Pakete hinweg auftreten. Diese Angriffe sind oft nur erkennbar, wenn man mehrere Pakete in einem bestimmten Zeitraum analysiert. In solchen Fällen kann es sinnvoll sein, Intervalle zu definieren, um ungewöhnliche Verhaltensweisen in der Abfolge der Pakete zu identifizieren. Diese Intervalle können festgelegt werden, um beispielsweise Verzögerungen zwischen Paketen, die Paketgröße oder das Senden von Paketen über einen bestimmten Zeitraum hinweg zu betrachten. Einen ersten Anhaltspunkt für die Größe der Intervalle kann durch die Masterarbeit von Wilhelm Schuhmacher [23] erlangt werden. Die genannte Arbeit behandelt auch das Thema Anomalieerkennung und setzt mitunter auf die gleichen Algorithmen. Dort wurde sich für eine Intervallgröße von einer Sekunde entschieden. Mit der Datenfusion (gerade über die Zeit) und der geringeren Laufzeit im Hinterkopf ist es tendenziell sinnvoller, noch kleinere Intervallgrößen zu untersuchen. Daher sind die untersuchten Größen auf eine Sekunde sowie auf eine Zehntel- und Zwanzigstel-Sekunde festgelegt. Diese Intervallgrößen sind dabei besonders auf die bandbreitenintensiven Datenströme ausgelegt. Wie in Abbildung 4.3 zu erkennen, liegen die längsten CAN-Zyklen bei zwei Sekunden, womit es auch Intervalle gibt, die keine Pakete enthalten. Der Aufwand, individuell für jeden Datenstrom eine geeignete Intervallgröße zu bestimmen, würde den Rahmen dieser Arbeit übersteigen.

Hat man sich entschieden, mit welcher Granularität das Netzwerk analysiert werden soll, muss geklärt werden, was analysiert wird. Der simpelste Ansatz umfasst Informationen über die Pakete: Start- und Zielpunkte, Protokolle, Paketgrößen etc. Außerdem ist es wie in der Arbeit von Soltani et al. [25] möglich, den Inhalt von Nachrichten zu analysieren und aus diesem Features zu generieren. Ein grundlegendes Problem besteht darin, dass die extrahierten Features schnell sehr hochdimensional werden. Um dieses Problem zu lösen, eignen sich Deep-Learning-Ansätze, da neuronale Netze oft in der Lage sind, hochkomplexe Muster in Daten zu erkennen.

Das Ziel der Datenaufbereitung ist es, die Lücke zwischen der Ausgabe der OMNeT++-Simulation und der Eingabe der Anomalieerkennung zu schließen und die Ergebnisse zu verbessern. Die OMNeT++-Simulation liefert für ein Szenario vier Netzwerkaufzeichnungen, eine für jeden Switch im Netzwerk.

Auf der anderen Seite steht die Anomalieerkennung, die in der Lage ist, eine Netzwerkaufzeichnung in gleichgroße Intervalle zu teilen und für jedes Intervall anhand von Bandbreite, durchschnittlicher Paketgröße, durchschnittlichem Paketabstand und dem Jitter zu bestimmen, ob es sich im Vergleich zu den Trainingsdaten um eine Anomalie handelt oder nicht. Außerdem kann entschieden werden, ob korrekt klassifiziert wurde, da sich in den Netzwerkaufzeichnungen ein Kommentar für jedes Paket befindet, der angibt, ob das Paket gültig oder ungültig ist und ob ein vorangegangenes Paket verloren gegangen ist. Da die Anomalieerkennung Intervalle bildet, erledigt sich die Frage, ob einzelne Pakete, Flows oder Intervalle betrachtet werden sollen. Weiterhin entfällt die Option, den Paketinhalt zu analysieren.

Die Ausgabe der Simulation und die Eingabe der Anomalieerkennung passen auf den ersten Blick zusammen. Doch es stellt sich die Frage, ob weitere Zwischenschritte der Datenaufbereitung dabei helfen können, die Ergebnisse zu verbessern. In jeder der vier Netzwerkaufzeichnungen finden sich eine Menge von UDP-Datenströmen und weitere Pakete, die bspw. zur Zeitsynchronisierung gehören. Aufgrund der schieren Anzahl von 216 Datenströmen ist es schwierig, Muster in der Gesamtkommunikation zu erkennen. Des Weiteren sind die verwendeten Netzwerkmetriken, die für die Intervalle ausgerechnet werden, nicht sehr präzise. Kleine Schwankungen fallen in der Gesamtmasse der Pakete kaum auf, da nur Durchschnittswerte berechnet werden. Wenn 1 von 100 Paketen in einem Datenstrom verloren geht, fällt es vielleicht auf, doch wenn eins von 1.000.000 Paketen in der Gesamtkommunikation verloren geht, ist es deutlich unwahrscheinlicher. Außerdem lässt sich nicht zurückverfolgen, in welchen Datenströmen eine Anomalie aufgetreten ist, was für den Umgang mit dieser wichtig ist. Und auch wenn Anomalien in verschiedenen Datenströmen auftauchen, die möglicherweise unterschiedliche Ursachen haben, kann nur erkannt werden, dass Anomalien vorliegen.

Aus diesen Gründen lässt sich leicht die Entscheidung treffen, die Datenströme im Netzwerk separiert zu betrachten.

4.4 Anomalieerkennung

Das in der Arbeit von Meyer et al. [19] vorgestellte Framework stellt einige Werkzeuge bereit. In dieser Sektion wird auf das Network Anomaly Detection System (NADS)

eingegangen und erläutert, wie es verwendet wird. Das NADS ist in ein in Python implementiertes Framework, das aus vier Schichten besteht: dem Stream Filtering, dem Metric Recording, dem Anomaly Detection Algorithm und dem Result Logging.

Das Stream-Filtering stellt die Eingabe in das Framework dar und hat die Aufgabe, auf die Daten aus einer der möglichen Eingabequellen (Netzwerkinterface, Simulation oder Netzwerkaufzeichnung) einen Filter anzuwenden und somit bspw. nur UDP-Pakete an die nächste Schicht weiterzugeben.

In der nächsten Schicht, dem Metric Recording, geht es dann um die sogenannte Feature Extraction. Dabei kann eine beliebige Kombination aus der Auswahl von vorgegebenen oder selbstdefinierten Metriken (Jitter, Bandbreite, durchschnittliche Größe eines Frames, durchschnittlicher Abstand zwischen Frames) ausgewählt werden. Die berechneten Metriken werden an die nächste Schicht weitergegeben.

Das Training und die Erkennung von Anomalien werden durch den Anomaly-Detection-Algorithmus durchgeführt. Auch hier können verschiedene vorgegebene Algorithmen verwendet werden, die aus der Python-Bibliothek scikit-learn[2] oder der Bibliothek TensorFlow[3] stammen, oder es können eigene Algorithmen hinzugefügt werden.

In der letzten Schicht, dem Result Logging, werden die Ergebnisse des Anomalieerkennungs-Algorithmus geloggt. Dazu gehören nicht nur die Vorhersagen der einzelnen Intervalle, sondern auch eine allgemeine Übersicht über die Performance in Form einer Confusion-Matrix.

[2]scikit-learn: Eine populäre Bibliothek für maschinelles Lernen in Python. [11]
[3]TensorFlow: Eine Open-Source-Bibliothek für maschinelles Lernen, entwickelt von Google. [10]

5 Umsetzung der Anomalieerkennung

Dieses Kapitel behandelt die im vorigen *Konzept* (Kapitel Kapitel 4) vorgestellten Schritte der Datenaufbereitung und Anomalieerkennung und lenkt dabei den Fokus auf die technische Umsetzung und Hürden, die in der Implementation aufgefallen sind.[1] Dabei werden weitere Überlegungen zum Laufzeitverhalten angestellt, um eine effektive Auswertung der Szenarien zu ermöglichen.

Die Modulstruktur des entwickelten Frameworks lässt sich in Abbildung 5.1 erkennen. Die Farben nehmen dabei Bezug auf die Abbildung 4.1, in der die fünf Phasen der Datenverarbeitung: Fahrzeugsimulation (grau), Datenstromseparierung (gelb), Berechnung der Netzwerkmetriken (orange), Anomalieerkennung (rot) und Datenfusion (lila) dargestellt werden. Die einzelnen Bestandteile der Software erfüllen jeweils eine spezifische Aufgabe, wobei sie u. A. auf externe Bibliotheken und Werkzeuge zurückgreifen. Um den Fokus auf die zugrundeliegende Architektur und die Beziehungen zwischen den Softwarebestandteilen zu lenken, zeigt das Diagramm die Modulstruktur des Projekts und explizit nicht alle implementierten Klassen. Außerdem wurde darauf verzichtet, sämtliche externe Abhängigkeiten aufzulisten und zu erläutern, da viele der Bibliotheken grundlegende Funktionalitäten wie beispielsweise das parsing von Dateien oder die Kommunikation mit dem Betriebssystem übernehmen.

Die Implementierung des gesamten Frameworks erfolgte mit Python. Grundlegend gibt es zwei Module, welche für die übergeordnete Verwaltung der Aufgaben verantwortlich sind: die Main Main und NADSController.

Die Klasse **Main** bezieht die Informationen, welche Aufgaben zu erledigen sind, aus einer Konfigurationsdatei im JSON-Format. Diese Datei umfasst u. A. folgende Informationen:

[1] Link zum Git-Repository, in dem sich der Code befindet: https://git.inet.haw-hamburg.de/mscherreiks/bachelor-martin-scherreiks (abgerufen am 05.05.2025).

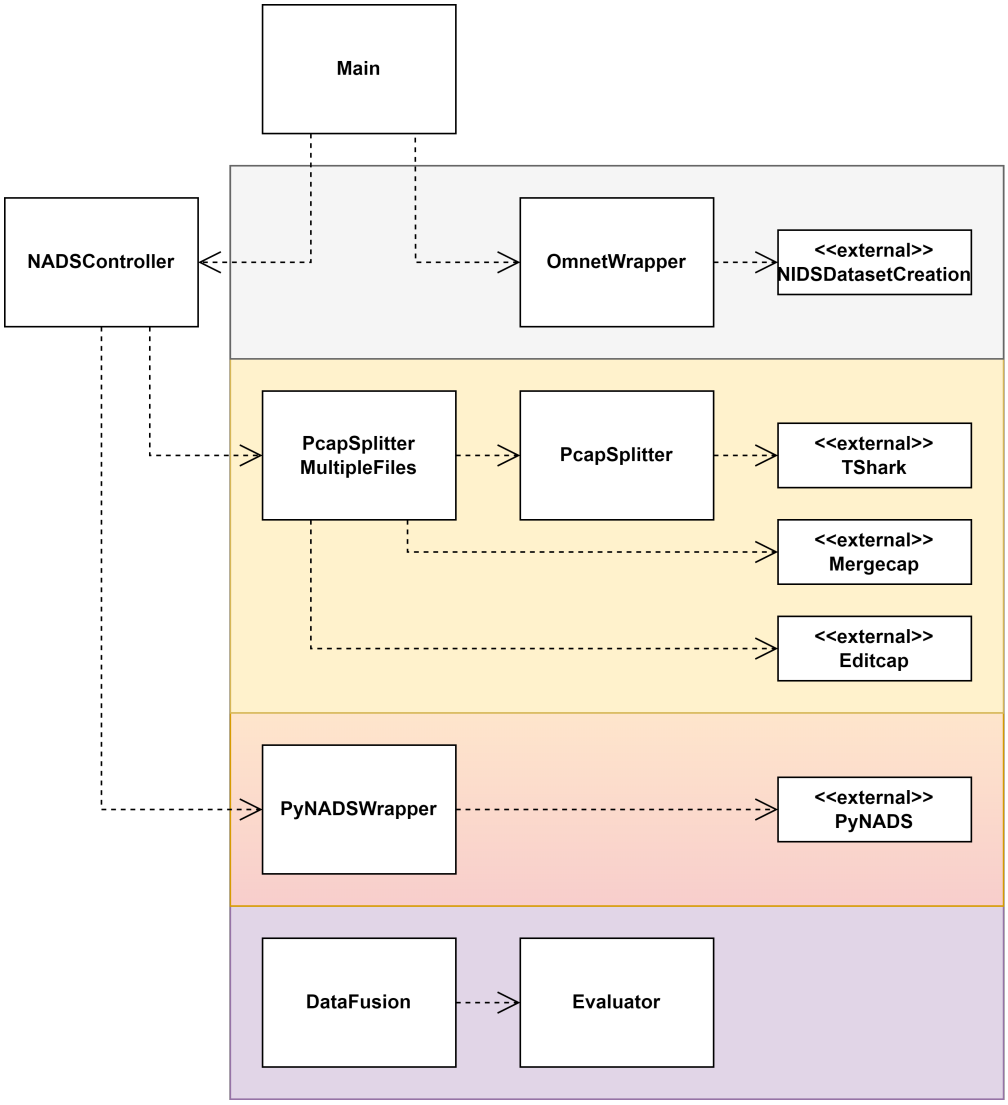


Abbildung 5.1: Die Abbildung zeigt die Modulstruktur des implementierten Frameworks und die Abhängigkeiten zu den genutzten externen Frameworks. Ein Modul besteht dabei aus einer oder mehreren Klassen. Die Farben beziehen sich auf die Schritte der Datenverarbeitung in Abbildung 4.1.

- Eine Liste von Anomaliewahrscheinlichkeiten, für die eine Simulation durchgeführt werden soll
- Eine Liste von Switches, für die Netzwerkaufzeichnungen vorliegen
- Eine Liste von Algorithmen, die die Auswertung der Netzwerkmetriken übernehmen sollen

- Eine Liste von Netzwerkmetriken, die berechnet werden sollen.
- Eine Liste von Intervallgrößen für die Berechnung der Netzwerkmetriken

Das **Main**-Modul berechnet aus den unterschiedlichen Anomaliewahrscheinlichkeiten im Szenario, den Switches und den Algorithmen alle möglichen Kombinationen. Für jede dieser Kombinationen wird dann eine Subkonfiguration erstellt. Zunächst wird mithilfe des OmnetWrappers überprüft, ob das entsprechend geforderte Anomalieszenario bereits als Netzwerkaufzeichnung vorhanden ist. Liegen die benötigten Aufzeichnungen nicht vor, so wird eine Omnet-Simulation gestartet, welche die Aufzeichnungen liefert. Anschließend werden die Subkonfigurationen an den NADS-Controller übergeben, der sich der Verwaltung der nachfolgenden Verarbeitungsschritte annimmt. Um auch bei einem Neustart des Programms den Überblick zu behalten, welche Aufgaben bereits erledigt worden sind, werden die Subkonfigurationen, die vom NADSController ausgeführt werden, mit einem Vermerk ‚temp' im Namen versehen, der durch den Vermerk ‚done' ersetzt wird, wenn die Konfiguration erfolgreich ausgeführt wurde. Dies sorgt dafür, dass dort weitergemacht werden kann, wo aufgehört wurde, wenn die Ausführung des Programms, etwa durch einen Neustart, unterbrochen wurde.

Der **NADSController** hat die Aufgabe, die Subkonfigurationen auszuführen, und unterteilt die Gesamtaufgabe dabei in viele kleine Teile. Die Anzahl der einzelnen Aufgaben bestimmt sich dabei durch zwei Faktoren: die Kombinationen von Netzwerkmetriken und Intervallgrößen sowie die verschiedenen Aufgabentypen, die für jede Kombination zu erledigen sind: die Datenstromseparierung, die Berechnung der Netzwerkmetriken für Trainings- und Testdaten sowie die Anomalieerkennung. Der NADSController arbeitet dabei in Phasen, da die einzelnen Schritte der Datenverarbeitung aufeinander aufbauen. Für eine möglichst effiziente Nutzung der verfügbaren Rechenressourcen wird ein Prozess-Pool verwendet. Dieser bekommt jeweils einzelne Task-Objekte, welche die Informationen über den Typ der Aufgabe (Datenstromseparierung usw.) und die jeweils benötigten Daten zur Aufgabe enthalten. Alle Tasks werden in einer sortierten Warteschlange gehalten, sodass die Phasen nacheinander durchlaufen werden können. Da die meisten Task-Objekte in ihrer Ausführung an den Punkt geraten, an dem sie ihre (Zwischen-)Ergebnisse in eine Datei schreiben, beginnt die Ausführung jeder Task damit, etwaige Dateien zu suchen, die dafür sprechen, dass das entsprechende Task-Objekt bereits teilweise oder ganz ausgeführt wurde. Dabei gilt das Prinzip: Wenn eine Datei sukzessive beschrieben wird, bekommt sie während der Ausführung den Vermerk

„temp“, der anschließend entfernt wird. Dies stellt sicher, dass Aufgaben nicht unnötigerweise mehrfach ausgeführt werden und Rückstände in Form von temporären Dateien entfernt werden können, um die Korrektheit der Ergebnisse zu schützen.

5.1 Fahrzeugsimulation

Bevor Daten verarbeitet werden können, müssen diese erstmal beschafft werden. Dies geschieht mithilfe des in OMNeT++ definierten Fahrzeugnetzwerks (siehe Abbildung 4.2), auf dem sowohl das Trainingsszenario als auch die Anomalieszenarien simuliert werden. Neben der Definition des Fahrzeugnetzwerks stammen auch die verschiedenen Arten von Anomalieszenarien aus dem NIDSDatasetCreation-Framework [19].

Die gesamte Kommunikation mit dem Framework und Omnet++ ist in der Omnet-Wrapper-Klasse gekapselt. Dieser bearbeitet die Konfigurationsdatei des Anomalieszenarios, in der bspw. die Wahrscheinlichkeit für das Auftreten einer Anomalie definiert ist. Je nachdem, welche Anomaliewahrscheinlichkeit benötigt wird, wird der Wert an den passenden Stellen der Konfigurationsdatei angepasst. Anschließend wird das NIDS-Dataset-Creation-Framework aufgerufen und auf das Ende der Simulation gewartet. Ist diese beendet, werden die Netzwerkaufzeichnungen in das gewünschte Verzeichnis verschoben, wo sie für die weitere Verarbeitung zur Verfügung stehen.

5.2 Datenstromseparierung

Diese Sektion befasst sich mit der Datenstromseparierung, also der Aufteilung der Netzwerkaufzeichnungen, in denen Anomalien erkannt werden sollen. Ausgegeben werden soll dabei eine Datei für jeden Datenstrom in der ursprünglichen Aufzeichnung. Dabei wird zunächst die technische Umsetzung und im Anschluss die Optimierung des Prozesses erläutert.

Die Phase der Datenstromseparierung wird durch zwei Klassen implementiert und erfolgt ihrerseits in zwei Phasen. Aufgrund der speziellen Anforderungen, die durch Paketkommentare und spezielle Ethernet-Erweiterungen in den Netzwerkwerkaufzeichnungen

entstehen, werden TShark[2] und die darauf aufbauenden Netzwerkwerkzeuge Mergecap[3] und Editcap[4] verwendet.

Diese können mit einer Vielzahl an Protokollen und Erweiterungen umgehen und sind auch in der Lage, Paketkommentare zu lesen und zu schreiben. Da diese Kommentare dem Zweck dienen, die Anomalien in den verschiedenen Szenarien zu markieren, sind Lesen und Schreiben eine Grundvoraussetzung.

In der ersten Phase, der Datenstromseparierung, müssen die Schlüssel ermittelt werden, anhand derer die verschiedenen Datenströme identifiziert und separiert werden können. Da das Netzwerk (siehe Abbildung 4.2) die Eigenschaft besitzt, dass alle Kombinationen von UDP-Start- und -Zielports eindeutig einen UDP-Stream identifizieren, können diese als Schlüssel verwendet werden. Mit diesem Wissen kann TShark aufgerufen werden, um für jedes Paket den Start- und Zielport auszugeben. Diese Liste kann dann auf einzigartige Elemente heruntergebrochen und als Grundlage für die nächste Phase verwendet werden. In dieser nächsten Phase wird ein Prozess-Pool verwendet, um für jede einzigartige Port-Kombination einen TShark-Aufruf zu starten. Dieser Aufruf enthält einen Filter, sodass die originale Datei nach allen Paketen durchsucht wird, die dem entsprechenden UDP-Stream angehören.

Diese Implementierung ist funktional, hat jedoch das Problem eines enormen Speicherverbrauchs bei zunehmender Größe der zugrundeliegenden Netzwerkaufzeichnung. Die virtuelle Maschine, auf der die Datenverarbeitung für diese Arbeit durchgeführt wurde, besitzt 32 Kerne und 128 GB Arbeitsspeicher. TShark verwendet so viel Arbeitsspeicher, dass bei der Datenstromseparierung einer 20 GB großen Aufzeichnung nur 8 Kerne gleichzeitig effizient arbeiten können. Den Swap-Space im Betriebssystem zu erhöhen (und damit Teile des Arbeitsspeichers zeitweise auf die Festplatte auszulagern) ist keine valide Option, da das System bei zunehmendem Verbrauch des Swap-Space immer langsamer wird. Die Vermutung liegt nahe, dass viele Page-Faults auftreten, da Tshark immer wieder auf verschiedene Teile des reservierten Speichers zugreift und so kein brauchbares Working-Set bestimmt werden kann. Um die gesamte Hardware-Kapazität nutzbar

[2] TShark ist das Kommandozeilen-Pendant zu Wireshark und dient der Analyse und dem Mitschneiden von Netzwerkverkehr [31].
[3] Mergecap erlaubt das Zusammenführen mehrerer Netzwerkaufzeichnungen in eine einzelne Datei [30].
[4] Editcap dient dem Bearbeiten von Netzwerkaufzeichnungen, z. B. zum Herausschneiden einzelner Pakete, Ändern von Zeitstempeln oder der Zerlegung in mehrere kleine Dateien [29].

zu machen, ist es notwendig, den Speicherverbrauch zu senken. Dieser verläuft in einem bestimmten, sägezahnartigen Muster. Im Laufe eines TShark-Aufrufs nimmt der Speicherverbrauch konstant zu. Wenn dieser beendet ist, wird der Speicher schlagartig freigegeben. Da TShark dieselbe Datei mit unterschiedlichen Filtern durchsucht, dauern die Aufrufe alle etwa gleich lang. Dieses Verhalten führt dazu, dass es große Spitzen im Speicherverbrauch gibt, wenn alle 32 Kerne gleichzeitig arbeiten und das Betriebssystem in letzter Konsequenz einzelne Prozesse beendet, um Speicher freizugeben. Eine Möglichkeit, den Speicherverbrauch zu senken, ist das Aufteilen der Netzwerkaufzeichnungen in kleinere Dateien. Die Aufteilung kann bspw. anhand einer festgelegten Anzahl von Paketen erfolgen, sodass in jeder Ausgabedatei maximal 1.000.000 Pakete vorliegen. Da TShark schneller mit der Aufteilung einer kleinen Datei fertig wird, ist auch die maximale Speicherauslastung geringer und zugleich die Anzahl der Tasks zunimmt. Dies sorgt dafür, dass die Spitzen im Sägezahn-Muster weniger stark ausgeprägt sind, dafür aber die Frequenz der Sägezähne zunimmt. Die kleineren Netzwerkaufzeichnungen werden dann getrennt in ihre Datenströme separiert und diese später zusammengefügt. Diese Aufgabe übernimmt das PcapSplitterMultipleFiles-Modul (siehe Abbildung 5.1). Zunächst wird in der „Divide“-Phase die Netzwerkaufzeichnung mit dem Editcap-Werkzeug [29] unterteilt. Anschließend wird die PcapSplitter-Klasse verwendet, um die Datenströme zu separieren. Mithilfe des Mergecap-Werkzeugs [30] werden die Teile der Datenströme in der „Merge“-Phase zu einer Datei zusammengeführt.

Die Ergebnisse der „Divide“- und „Merge“-Phasen sprechen für sich und sind in Abbildung 5.2 zu sehen. Dort lässt sich erkennen, dass sich die Spitzen im Speicherverbrauch um ca. 50% reduziert haben und mit einer Reduzierung von ca. 12550 auf 4005 Sekunden ungefähr zwei Drittel der Laufzeit eingespart werden können.

5.3 Berechnung der Netzwerkmetriken

Für die Berechnung der Netzwerkmetriken wird das PyNADS-Framework verwendet, welches in der Arbeit von Meyer et al. [19] vorgestellt wird. Dieses Framework ist in der Lage, Netzwerkmetriken auszurechnen, diese zu speichern und hinterher als Trainingsdaten für die Anomalieerkennung zu nutzen. Dabei wird in dieser Arbeit u. A. der Einfluss verschiedener Metriken, die für Intervalle unterschiedlicher Größen berechnet und von verschiedenen Anomalieerkennungs-Algorithmen auf Anomalien geprüft werden, untersucht. Da die Anomalieerkennung mindestens zwei der vier zur Verfügung

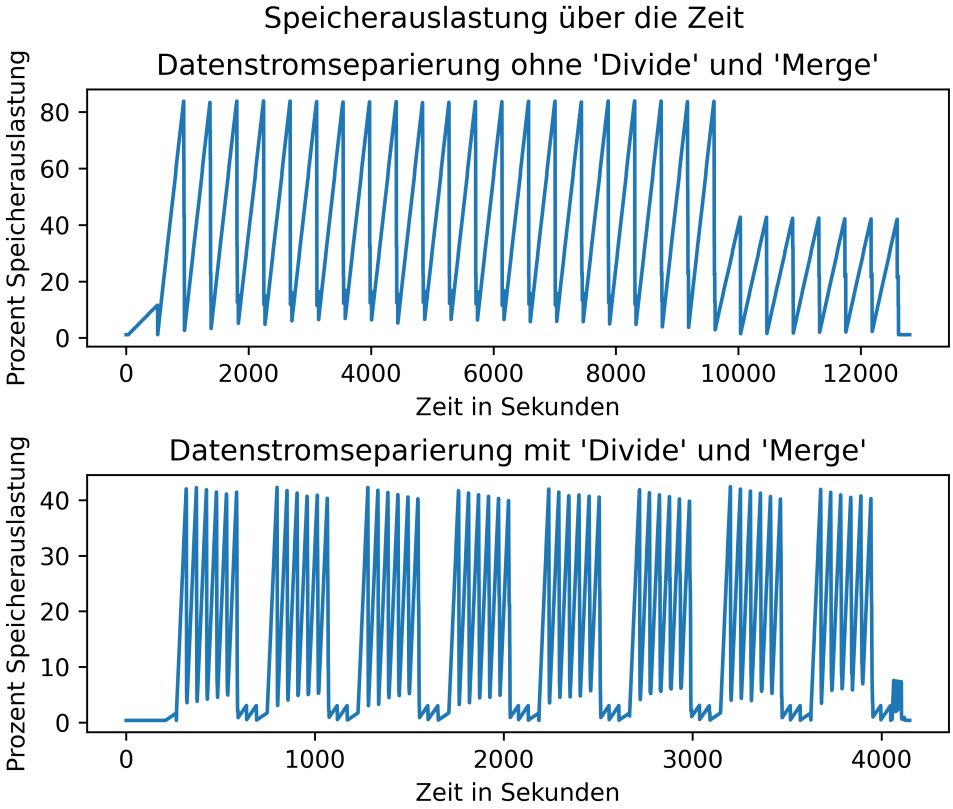


Abbildung 5.2: Die Abbildung zeigt die prozentuale Speicherbelastung über die Zeit im Vergleich zwischen der ursprünglichen Datenstromseparierung und jener mit „Divide“- und „Merge“-Phase bei einer Netzwerkaufzeichnung von ca. 22 GB.

stehenden Metriken benötigt (Bandbreite, Jitter, durchschnittliche Größe eines Frames und durchschnittlicher Abstand zwischen Frames), gibt es bei den vier verschiedenen Netzwerkmetriken insgesamt elf einzigartige Kombinationen. Wenn wir davon ausgehen, dass drei Intervallgrößen untersucht werden, dann gibt es bei sechs Algorithmen und je ca. 200 Datenströmen in den vier Netzwerkaufzeichnungen (eine Aufzeichnung pro Switch im Netzwerk (siehe Abbildung 4.2)) insgesamt $200 \cdot 4 \cdot 6 \cdot 3 \cdot 11 = 158.400$ verschiedene Kombinationen pro Anomalie-Szenario, für die eine Anomalieerkennung durchgeführt werden soll. Da die Netzwerkmetriken sowohl für das Training der Anomalieerkennung als auch für die Durchführung berechnet werden müssen, verdoppeln sich die benötigten Berechnungen auf 316.800. Bei einer durchschnittlichen Datenstromgröße von $\frac{55\,\mathrm{GB}}{4 \cdot 200} = 68,75\,\mathrm{MB}$ ergeben sich $316800 \cdot 68,75 = 21,78\,\mathrm{TB}$ an zu verarbeitenden Da-

ten allein für die Berechnung der Netzwerkmetriken für ein Anomalie-Szenario. Aufgrund dieser Datenmenge stellt sich die Frage, ob es Einsparpotentiale gibt.

Im Grunde genommen müssen für viele der Kombinationen aus Intervall, Algorithmus usw. die gleichen Netzwerkmetriken ausgerechnet werden. So ist es egal, welcher Algorithmus die Anomalieerkennung ausführt, da sich die Netzwerkmetriken nicht verändern. Außerdem reicht es aus, wenn alle Metriken einmal für alle Datenströme berechnet und als Tabelle gespeichert werden, weil die verschiedenen Untermengen von Metriken aus der Tabelle abgelesen werden können, indem nur die benötigten Spalten betrachtet werden. Somit werden aus 158.400 Kombinationen $200 \cdot 4 \cdot 3 = 2.400$, während sich die Datenmenge auf $2.400 * 68,75$ MB $= 165$ GB bzw. auf ca. 0,76 % der ursprünglichen 21,78 TB reduziert.

Das PyNADS-Framework selbst ist dabei nur in der Lage, die Netzwerkmetriken für das Training der Algorithmen zu speichern, nicht aber für die Testdaten der Anomalieerkennung selbst. Außerdem werden in der Datei nur die notwendigen Metriken gespeichert, nicht aber alle verfügbaren, was dazu führt, dass das Framework selbst nur bedingt zur Einsparung in der Lage ist. Daher ist der NADSController neben seiner Prozessmanagement-Funktion auch für die Erstellung von temporären Dateien verantwortlich. In diese Dateien werden die jeweils benötigten Netzwerkmetriken geschrieben. Hierfür werden alle verfügbaren Netzwerkmetriken geladen und nur die benötigten Spalten in die temporäre Datei geschrieben. Um das Framework darüber hinaus auch in die Lage zu versetzen, die Netzwerkmetriken für die Testdaten speichern und laden zu können, wurden einige Änderungen vorgenommen. Im Ablauf des PyNADS-Frameworks werden nach und nach die Pakete aus den Netzwerkaufzeichnungen gelesen und anhand der Timestamps einem Intervall zugeordnet. Immer wenn ein Intervall fertig geladen und die Metriken berechnet wurden, werden diese Informationen an den aktiven Anomalieerkennungs-Algorithmus gegeben, welcher bestimmt, ob es sich um eine Anomalie handelt oder nicht. An dieser Stelle wurde eine Anpassung vorgenommen, sodass die berechneten Metriken zusammen mit dem Timestamp des Intervalls in eine Datei geschrieben werden. Neben den Timestamps und den Metriken werden außerdem die Informationen aus den Paket-Kommentaren gespeichert, die verwendet werden, um zu bestimmen, ob der Algorithmus die korrekte Klassifikation getroffen hat. Diese Kommentare sind für die Evaluation der Erkennungsleistung erforderlich. Die gespeicherten Daten können im nächsten Durchlauf das Lesen der Pakete ersetzen, sodass die Anomalieerkennung nicht mehr auf die Netzwerkaufzeichnungen selbst zurückgreifen muss.

5.4 Anomalieerkennung

Die Anomalieerkennung stellt den zentralen Punkt in der Datenverarbeitung dar. Hier laufen die berechneten Netzwerkmetriken der Trainings- und Testdatensätze zusammen, wie in der Abbildung 4.1 aus dem *Konzept-Kapitel* (Kapitel 4) zu erkennen ist. In diesem Schritt können die verschiedenen implementierten Algorithmen auf Grundlage der Trainingsdaten trainiert und anschließend auf die Netzwerkmetriken der simulierten Anomalieszenarien angewendet werden. Quelle der Algorithmen ist das PyNADS-Framework, welches in der Arbeit von Meyer et al. [19] vorgestellt wurde. Um die Integration des Frameworks in den Gesamtprozess zu gewährleisten, wird wie im vorangegangenen Schritt der PyNADS-Wrapper eingesetzt. Dieser Wrapper übernimmt die Kommunikation mit dem PyNADS-Framework, wobei er sicherstellt, dass alle relevanten Parameter und Optionen korrekt an PyNADS übergeben werden und der Klassifikationsprozess nahtlos in die bestehende Verarbeitungspipeline integriert ist.

Für die Verwaltung und Parallelisierung der durchzuführenden Klassifikationen ist auch in diesem Schritt der zentrale NADSController vorgesehen. Während des Tests der Implementation ist dabei aufgefallen, dass einer der nutzbaren Algorithmen deutlich schlechter parallelisierbar war als die anderen. Dies ist durch den Umstand aufgefallen, dass die Anwendung des MS-Algorithmus auf einen Datenstrom nach wenigen Sekunden abgeschlossen war, wenn dieser Algorithmus allein auf dem System lief. Im Gegensatz dazu hat die Ausführung mehrere Minuten in Anspruch genommen, wenn alle Prozessoren des Systems für unterschiedliche Instanzen genutzt wurden. Auffällig war die Anzahl der Threads auf dem Gesamtsystem, sobald das parallelisierte Programm ausgeführt wurde. Da aktiv nur an einer Stelle parallelisiert wurde, lag die Vermutung nahe, dass es eine Bibliothek geben muss, die selbst im Hintergrund parallelisiert. Nach einiger Suche ist der Verdacht dann auf die Bibliotheken MKL[5] und OpenBLAS[6] gefallen. Diese Bibliotheken sind dafür bekannt, im Hintergrund zu parallelisieren, um schnelle mathematische Operationen zu ermöglichen. Über einen Betriebssystemaufruf kann die Anzahl der erzeugten Threads jedoch auf eins gesetzt werden, was das Problem der ineffizienten Parallelisierung gelöst hat.

[5] https://www.intel.com/content/www/us/en/developer/tools/oneapi/onemkl.html Intel Math Kernel Library (MKL) ist eine hochoptimierte mathematische Bibliothek für Vektor-, Matrix- und lineare Algebra-Operationen.

[6] https://www.openblas.net/ OpenBLAS ist eine optimierte BLAS-Implementierung (Basic Linear Algebra Subprograms), die für schnelle Matrixoperationen genutzt wird.

Ein weiteres Problem bestand bei der Anomalieerkennung mit dem EE-Algorithmus. Versucht man den Algorithmus auf die separierten Datenströme anzuwenden, bricht die Ausführung teilweise mit der Fehlermeldung ab, dass die Kovarianzmatrix singulär sei. Das Problem liegt in der Varianz in den betroffenen Datenströmen, da die berechneten Netzwerkmetriken der Intervalle teilweise identisch sind. Da der Algorithmus nur mit gültiger Kovarianzmatrix ausgeführt werden kann, die Varianz der betroffenen Datenströme aber nicht beeinflusst werden kann, ist der EE-Algorithmus für die Anomalieerkennung in Fahrzeugnetzwerken ungeeignet.

6 Evaluation der Anomalieerkennung

In diesem Kapitel sollen die Ergebnisse der Anomalieerkennung evaluiert werden. Die vollständigen Tabellen mit allen Ergebnissen finden sich im Anhang. Insgesamt wurden vier Szenarien durchgeführt, die jeweils unterschiedliche Angriffe und ein unterschiedliches Ziel haben. Die maximalen Anomaliewahrscheinlichkeiten stammen aus der Masterarbeit von Wilhelm Schuhmacher [23]. In dieser Arbeit wurde die Anomalieerkennung in einem ähnlichen Kontext in realer Hardware umgesetzt. Da diese Wahrscheinlichkeiten (zwischen 50 und 100%) für sehr viele Anomalien sorgen, wurden alle Szenarien zusätzlich mit zwei geringeren Anomaliewahrscheinlichkeiten simuliert, um die Grenzen der Erkennungsleistung besser einschätzen zu können. Alle betroffenen Datenströme gehören zu denjenigen im Netzwerk (siehe Abbildung 4.2), die am meisten Bandbreite besitzen und nach dem Standard 802.1CB [16] redundante Wege nutzen. In den folgenden Sektionen sollen zunächst die vier Szenarien und zusätzlich ein Baseline-Szenario ohne Anomalien vorgestellt und die individuellen Erkenntnisse beleuchtet werden. Im letzten Teil dieses Kapitels werden die Ergebnisse im Kontext der Datenfusion zusammengefasst und bewertet.

6.1 Baseline

Das Baseline-Szenario muss getrennt von den anderen betrachtet werden, denn es handelt sich um die Auswertung des normalen Datenverkehrs im Fahrzeugnetzwerk (siehe Abbildung 4.2). Es dient als ein Blick auf die Leistung der Algorithmen, wenn es überhaupt keine Anomalien gibt. Die Metriken Präzision, Recall und F1 sind 0, da sie sich immer auf die TN beziehen, von denen es ohne Anomalien keine geben kann. Sehr interessant sind der AE und der SVM-Algorithmus, da diese Algorithmen durchweg eine sehr hohe FPR besitzen. Auf der anderen Seite stehen die Algorithmen HBO und MS, die sehr wenig FPs produziert haben.

Tabelle 6.1: Ergebnisse der Anomalieerkennung für das Baseline-Szenario. In diesem Szenario sind keine Anomalien aufgetreten, weswegen Precision, Recall und F1-Score 0 sind. Für jede Kombination wird jeweils nur die beste Algorithmus-Konfiguration angezeigt..

Kombination					**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Metriken**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.0	0.05	AE	None: 0.0	FG-J	0	0	0	0.13226	138670.5
0.0	0.1	AE	None: 0.0	FS-FG-J	0	0	0	0.14475	192744
0.0	1.0	AE	None: 0.0	B-FS-FG-J	0	0	0	0.1432	274350
0.0	0.05	HBO	Cont.: 0.0001	FS-J	0	0	0	**0.00012**	130.5
0.0	0.1	HBO	Cont.: 0.0001	FS-J	0	0	0	**0.00023**	300
0.0	1.0	HBO	Cont.: 0.0001	B-FS	0	0	0	**0.00243**	4650
0.0	0.05	IF	Cont.: 0.0001	B-FS	0	0	0	**0.0001**	106.5
0.0	0.1	IF	Cont.: 0.0001	B-FS	0	0	0	**0.00013**	171
0.0	1.0	IF	Cont.: 0.0001	B-FS	0	0	0	**0.00025**	480
0.0	0.05	KM1	BEF: 0.8	FS-J	0	0	0	0.01698	17808
0.0	0.1	KM1	BEF: 0.8	FS-J	0	0	0	0.02962	39441
0.0	1.0	KM1	BEF: 0.8	B-FS-FG	0	0	0	0.13847	265290
0.0	0.05	MS	BEF: 2.0	FS-J	0	0	0	**0**	**0**
0.0	0.1	MS	BEF: 2.0	FS-J	0	0	0	**0.00001**	9
0.0	1.0	MS	BEF: 2.0	B-J	0	0	0	**0.00041**	780
0.0	0.05	SVM	Cont.: 0.0001	B-FS-FG-J	0	0	0	0.64386	675082.5
0.0	0.1	SVM	Cont.: 0.0001	B-FS-J	0	0	0	0.89568	1192623
0.0	1.0	SVM	Cont.: 0.0001	B-J	0	0	0	1	1915830

6.2 Eliminate Auto Brake

Das Eliminate Auto Brake (EAB)-Szenario betrifft den Datenstrom im Netzwerk (siehe Abbildung 4.2), der für das automatische Bremsen des Fahrzeugs verantwortlich ist. Der Datenstrom beginnt beim ADAS und wird von dort über die hinteren und vorderen Switches an die Zonal-Controller weitergegeben. Der simulierte fehlerhafte Link-Layer, der für die Anomalien verantwortlich ist, beeinflusst den Datenstrom zwischen den Switches hinten rechts und dem vorne rechts. Da nur der Ausgang betroffen ist und sich der Datenstrom auf den zwei möglichen Wegen (links und rechts) redundant durch das Netzwerk bewegt, sind (bis auf den Switch hinten rechts) alle Aufzeichnungen der Switches betroffen [20].

Es gibt drei untergeordnete Szenarien, die jeweils unterschiedliche Wahrscheinlichkeiten für das Auftreten einer Anomalie besitzen: 10%, 25% und 50%. Diese Wahrscheinlichkeiten beziehen sich jeweils auf die Eliminierung eines Pakets am o. g. Switch.

Die Ergebnisse der Anomalieerkennung im EAB-Szenario sind in der Tabelle A.2 aufgelistet. Zunächst fallen zwei Algorithmen auf, die sehr gute Leistungen erzielt haben. Der HBO-Algorithmus hat mit einem minimalen Recall-Wert von ca. 99,5 % immer nahezu alle Anomalien erkannt. Bei einer Intervallgröße von 0,05 Sekunden konnte der Algorithmus außerdem eine Präzision von ca. 95 % erreichen. Ähnlich gute Ergebnisse konnte auch der MS-Algorithmus erzielen, während der SVM und AE durchweg sehr hohe FPRs erzielten. Der IF-Algorithmus hat so gut wie keine der Anomalien erkannt, während der KM-Algorithmus mit einem Cluster sehr mittelmäßige Leistungen erbracht hat. Insgesamt scheint eine Intervallgröße von einer Sekunde für deutlich schlechtere Ergebnisse zu sorgen.

6.3 Inject LIDAR

Das Inject Lidar (IL)-Szenario betrifft den Datenstrom im Netzwerk (siehe Abbildung 4.2), der die Daten des LIDAR-Sensors vorne-links durch das Netzwerk transportiert. Der Datenstrom beginnt dementsprechend beim LIDAR und wird von dort über die beiden direkt verbundenen Switches an das ADAS gesendet. Der simulierte fehlerhafte Link-Layer, der für die Anomalien verantwortlich ist, beeinflusst den Datenstrom zwischen den beiden oberen Switches. Da die Daten des Sensors vom Switch oben-links über zwei Routen versendet werden, von denen nur diejenige, die im Uhrzeigersinn verläuft, betroffen ist, lassen sich die Anomalien in den Aufzeichnungen der Switches vorne-rechts und hinten-links erkennen [20].

Es gibt drei untergeordnete Szenarien, die jeweils unterschiedliche Wahrscheinlichkeiten für das Auftreten einer Anomalie besitzen: 25%, 50% und 100%. Diese Wahrscheinlichkeiten beziehen sich jeweils auf das Einschleusen eines Pakets in den Datenstrom am o. g. Switch in Intervallabständen von 10 ms. Damit wird bei einer 100-prozentigen Wahrscheinlichkeit alle 10 ms ein zusätzliches Paket in den Datenstrom eingeschleust.

Die Ergebnisse der Anomalieerkennung im IL-Szenario sind in der Tabelle A.3 aufgelistet. Zunächst fällt auf, dass nur ein Algorithmus gute Leistungen erzielen konnte. Der IF-Algorithmus hat unter den richtigen Voraussetzungen, einer Intervallgröße von 0,05 Sekunden, 87-97% der Anomalien erkennen können, ohne dabei viele FPs hervorzurufen. Die anderen Algorithmen können zwar mit guten Recall-Werten überzeugen, bei der Präzision gibt es im Gegensatz dazu aber viel Verbesserungspotenzial.

6.4 Delay Manual Steer

Das Delay Manual Steer (DMS)-Szenario betrifft den Datenstrom im Netzwerk (siehe Abbildung 4.2), der für das manuelle Lenken des Fahrzeugs verantwortlich ist. Der Datenstrom beginnt beim Zonal-Controller vorne-links und wird von dort über die Switches an alle anderen Zonal-Controller weitergegeben. Der simulierte fehlerhafte Link-Layer, der für die Anomalien verantwortlich ist, beeinflusst den Datenstrom zwischen den Switches vorne-links und vorne-rechts. Da nur der Ausgang betroffen ist und sich der Datenstrom auf den zwei möglichen Wegen (links und rechts) redundant durch das Netzwerk bewegt, sind (bis auf den Switch vorne-links) alle Aufzeichnungen der Switches betroffen [20].

Es gibt drei untergeordnete Szenarien, die jeweils unterschiedliche Wahrscheinlichkeiten für das Auftreten einer Anomalie besitzen: 10%, 25% und 50%. Diese Wahrscheinlichkeiten beziehen sich jeweils auf die Verzögerung eines Pakets um 10 µs.

Die Ergebnisse der Anomalieerkennung im DMS-Szenario sind in der Tabelle A.4 aufgelistet. Auf den ersten Blick fallen die insgesamt eher schlechten Ergebnisse auf. So haben die meisten Algorithmen nicht alle Anomalien erkannt. Nur der SVM-Algorithmus konnte mit einem perfekten Recall überzeugen. Aufgrund der enorm hohen FPR von bis zu 100% sind die Ergebnisse aber kaum von Wert. Nur die HBO- und MS-Algorithmen konnten relativ gute Recall- und Präzisionswerte um 90% bzw. 45% hervorbringen.

6.5 Reorder Camera

Das Reorder Camera (RC)-Szenario betrifft den Datenstrom im Netzwerk (siehe Abbildung 4.2), der für den Versand der vorderen Kamera-Daten verantwortlich ist. Der Datenstrom beginnt bei der vorderen Kamera und wird von dort über alle Switches an das ADAS weitergegeben. Der simulierte fehlerhafte Link-Layer, der für die Anomalien verantwortlich ist, beeinflusst den Datenstrom zwischen den Switches vorne und hinten rechts. Da nur der Ausgang betroffen ist und sich der Datenstrom auf den zwei möglichen Wegen (links und rechts) redundant durch das Netzwerk bewegt, ist nur die Aufzeichnung des Switches hinten rechts von den Anomalien betroffen [20].

Es gibt drei untergeordnete Szenarien, die jeweils unterschiedliche Wahrscheinlichkeiten für das Auftreten einer Anomalie besitzen: 10%, 25% und 50%. Diese Wahrscheinlichkei-

ten beziehen sich jeweils auf die Entfernung und Wiedereingliederung eines Pakets nach dem direkt folgenden Paket.

Die Ergebnisse der Anomalieerkennung im RC-Szenario sind in der Tabelle A.5 aufgelistet. Insgesamt hatten die Algorithmen Probleme mit dem Szenario. Hohe Recall-Werte kommen hier mit schlechten Präzisionswerten einher. Die AE– und SVM–Algorithmen stechen hierbei zusätzlich mit hohen FPRs hervor.

6.6 Zwischenfazit

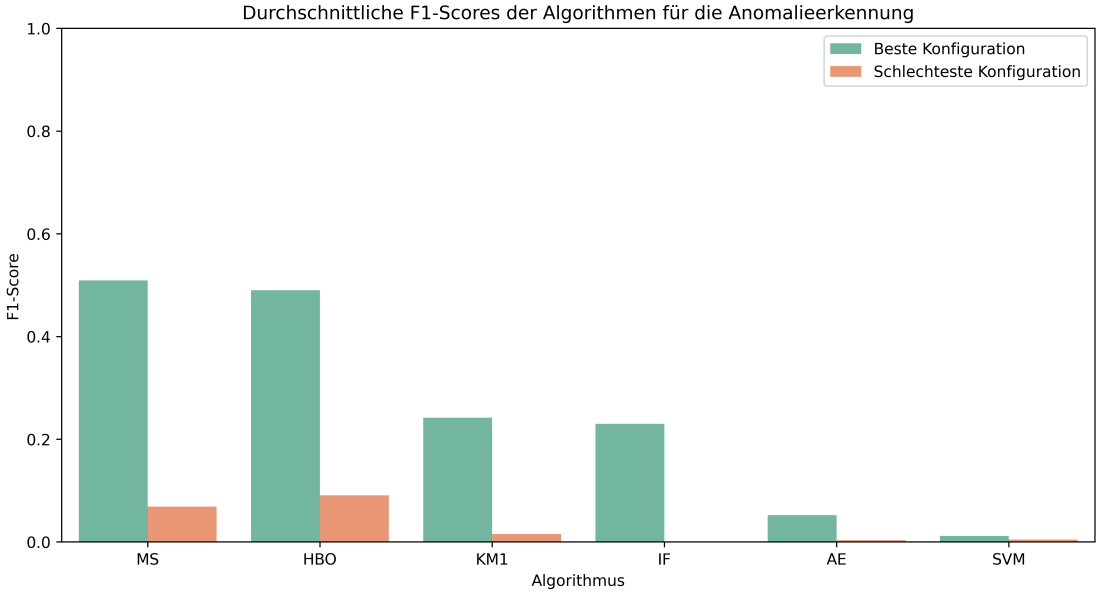


Abbildung 6.1: Die Abbildung zeigt die durchschnittlichen F1-Scores der Algorithmen in der Anomalieerkennung. Dargestellt sind jeweils die beste und die schlechteste Konfiguration.

Nach der Evaluation der einzelnen Szenarien lässt sich festhalten, dass die Ergebnisse sehr durchwachsen sind. In den Ergebnissen findet sich nur eine Konstante: Die FPs pro Stunde überschreiten überall ein vertretbares Maß, sodass in den meisten Fällen weit mehr als eine Falschmeldung pro Sekunde zu erwarten wäre. Abgesehen davon ist auffällig, dass die Algorithmen in allen Szenarien ähnliche FPRs aufweisen. Dieser Umstand lässt sich dadurch erklären, dass ein Großteil des Datenverkehrs in den verschiedenen Szenarien sehr ähnlich ist, da immer nur ein einziger Datenstrom betroffen ist, und die FPR (zusammen mit den FP/h) die einzige Metrik ist, die unabhängig von

den korrekt identifizierten Anomalien ist. Neben dieser erwartbaren Beobachtung fällt auf, dass die Algorithmen insgesamt am besten im EAB abgeschnitten haben. Besonders hat sich außerdem der IF-Algorithmus verhalten, der ausschließlich im IL-Szenario sehr gute Leistungen erzielt hat. In allen anderen Szenarien wurden nahezu keine Anomalien erkannt.

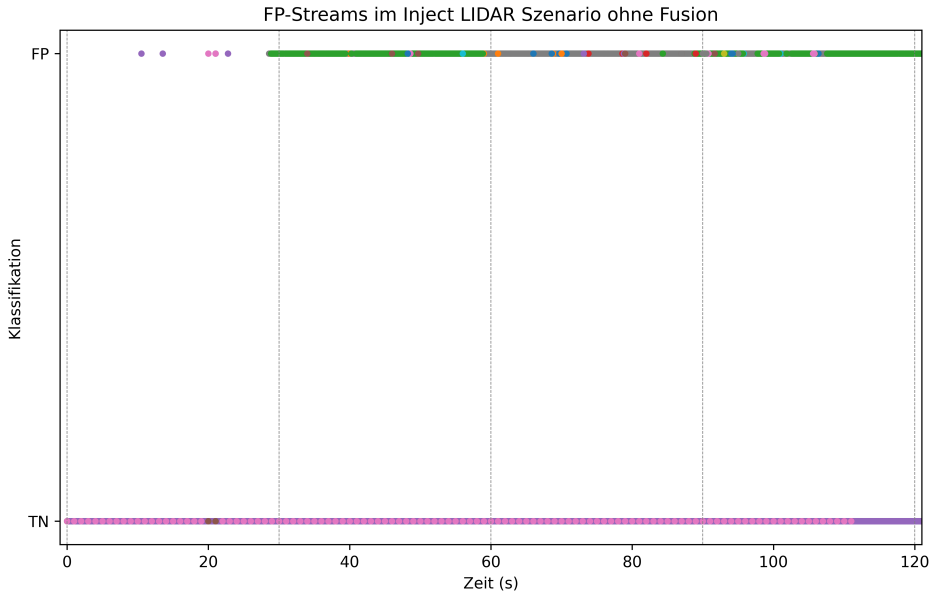


Abbildung 6.2: Die Abbildung zeigt alle Streams im IL-Szenario, in denen FPs aufgetreten sind. Die Anomalieerkennung wurde mit dem MS-Algorithmus mit der besten durchschnittlichen Konfiguration durchgeführt. Aufgrund sehr vieler Datenströme wurde die Legende weggelassen

Für die Datenfusion lässt sich der Schluss ziehen, dass die beiden Algorithmen AE und SVM nicht für die weitere Verarbeitung geeignet sind, da die FPR durchweg zu hoch ist.

Weiterhin kann der Schluss gezogen werden, dass der Recall-Wert leicht durch eine Fusion über die Zeit verbessert werden kann, da oft über 80% und damit ein Großteil der Anomalien erkannt werden konnten. Das größte Problem liegt einerseits in der hohen Zahl der FPs und andererseits in den sehr wechselhaften Leistungen. Das Ziel der Datenfusion ist, eine möglichst universelle Lösung zu finden, die in allen Szenarien gute Ergebnisse liefert. Dies ist bspw. für den Einsatz des IF-Algorithmus verheerend, da

dieser ausschließlich in einem Szenario überzeugen konnte. Da die meisten Algorithmen kein Problem mit dem Erkennen von Anomalien hatten, kann die Datenfusion ein hohes Vertrauen in eine ‚normale' Klassifizierung legen. So besteht die Hoffnung, dass die Algorithmen in verschiedenen Situationen FPs erzeugen und die Fusion für eine erhebliche Steigerung der Präzision sorgen könnte, da nur wenige Stimmen gegen eine Anomalie-Klassifizierung ausreichen würden, um ein FP zu verhindern.

Bezüglich des Auftretens von FPs zeigt die Abbildung stellvertretend für alle Szenarien 6.2 die Datenströme im IL-Szenario, in denen FPs vorkommen. Die Anomalieerkennung wurde mit der besten durchschnittlichen Konfiguration des MS-Algorithmus vorgenommen. Der Abbildung lässt sich kein Muster entnehmen, da die FPs (abgesehen vom ersten Viertel) gleichmäßig über die Zeit verteilt scheinen.

7 Konzept der Datenfusion

In diesem Kapitel sollen grundlegende, konzeptionelle Überlegungen für die Datenfusion angestellt werden. Dabei geht es vor allem darum, eine Auswahl von Fusionsverfahren vorzustellen und im Anschluss die Kombination der Verfahren zu erläutern. Außerdem ist es aber von entscheidender Bedeutung, zu spezifizieren, auf welchen Daten die Fusion konkret aufbaut und wie die Fusion allgemein abläuft.

7.1 Fusionsverfahren

In dieser Sektion soll zunächst ein kleiner Überblick über mögliche Fusionsverfahren gegeben werden. Anschließend werden die Verfahren vorgestellt werden, die für diese Arbeit implementiert werden sollen. Dabei sind die Mehrheitsfusion und die D-S Evidenztheorie für die primäre Fusion gedacht. Die Fusion über die Zeit hingegen eignet sich nur als sekundäre Fusion, die auf den Ergebnissen der primären Fusion basiert.

Auswahl der Verfahren

Es gibt viele etablierte Verfahren, die für die Datenfusion verwendet werden können. Der simpelste Ansatz ist dabei ein Fusionsverfahren, welches mit Mehrheiten oder Grenzwerten arbeitet. Die prozentuale Gewichtung der zugrundeliegenden Quellen entscheidet dabei über das Ergebnis der Fusion. Außerdem können sogenannte Fuzzy-Logic-Module genutzt werden, die mit if-then-Regeln arbeiten, um das Fusionsergebnis zu bestimmen, wie es in der Arbeit [34] getan wurde. Vielversprechend ist außerdem das D-S-Verfahren, welches in der Arbeit [32] für die Datenfusion eingesetzt wurde. Dieses arbeitet mit Massenfunktionen, welche die Meinungen der Datenquellen widerspiegeln sollen und zu einer gemeinsamen Massenfunktion (also einer gemeinsamen Überzeugung) kombiniert werden können. Grundsätzlich anders ist die Fusion über die Zeit, die nicht verschiedene Sensoren fusioniert, sondern dafür eingesetzt werden kann, das Ergebnis im zeitlichen Kontext

zu glätten. Zuletzt können neuronale Netzwerke in verschiedenen Ausprägungen (Convolutional, Fuzzy, Deep) verwendet werden. Ein Beispiel bietet hierbei die Arbeit [18]. Neuronale Netze haben den Nachteil, dass sie auf qualitative Trainings- und Testdaten angewiesen sind, um gute Ergebnisse zu erzielen. Da die Beschaffung dieser Daten sehr aufwendig ist, wird dieses Verfahren im Rahmen dieser Arbeit außen vorgelassen. Neben den neuronalen Netzwerken sind auch Fuzzy-Logic-Module nicht Teil dieser Arbeit, da sie aufgrund der starren Struktur der if-then-Regeln eher weniger für die Anomalieerkennung geeignet sind, da ein Anomaliedetektor im Idealfall auch mit unbekannten Angriffen zurechtkommen sollte. Die übrigen Verfahren werden im Folgenden vorgestellt.

Mehrheitsfusion

Eine der grundlegenden Methoden zur Fusion ist die Mehrheitsfusion. Dabei wird überprüft, ob in einem Intervall eine vordefinierte Anzahl bzw. ein Anteil der zur Verfügung stehenden Anomalieerkennungs-Instanzen Anomalien erkannt hat. Dieser Ansatz kann durch Gewichtungen verfeinert werden, indem der Grenzwert nach oben oder unten verschoben wird. Dies kann insbesondere in Situationen mit vielen False Positives oder False Negatives hilfreich sein, um die Erkennungsleistung zu verbessern.

Dempster-Shafer Evidenztheorie

Die D-S Evidenztheorie [24] bezeichnet eine mathematische Theorie, welche als Generalisierung der Wahrscheinlichkeitstheorie aufgefasst werden kann. Die Wahrscheinlichkeiten finden sich in der Evidenztheorie bezogen auf Mengen von Events wieder und werden auch als Massen bezeichnet. Im Gegensatz zur Wahrscheinlichkeitstheorie kann sich die Evidenz demnach auf mehr als ein Event gleichzeitig beziehen. Ein weiterer entscheidender Unterschied liegt in der fundamental inhärenten Darstellung von Unsicherheiten, sodass auch weniger präzise Inputs verarbeitet werden können. Grundlegend existieren dabei drei wichtige Funktionen: Zunächst gibt es die Probability-Assignment-Function oder Massenfunktion, die jeder Menge in der Potenzmenge der möglichen Events eine Masse zwischen 0 und 1 zuweist. Die zugewiesene Masse $m(A)$ kann als Evidenz für die Menge A angesehen werden, wird von einigen Forschenden aber auch als Wahrscheinlichkeit interpretiert. Die Summe der zugewiesenen Massen aller Mengen in der Potenzmenge ergibt dabei stets 1. Die zweite wichtige Funktion wird als Belief bezeichnet und stellt

die Summe aller Submengen für eine Menge dar. Sie kann im Kontext der Anomalieerkennung auch als Unterstützung für eine Hypothese A interpretiert werden:

$$\mathrm{Bel}(A) = \sum_{B|B\subseteq A} m(B)$$

Die Plausibilität einer Menge A beschreibt den Wert der Massen, die eine Hypothese A nicht ausschließen:

$$\mathrm{Pl}(A) = \sum_{B\,|\,B\cap A\neq\emptyset} m(B)$$

Interessant wird es, wenn mehrere Massenfunktionen miteinander kombiniert werden sollen. Man kann diesen Vorgang auch als Fusion von verschiedenen Sensorquellen aufgreifen. Die Dempster-Shafer-Regel bietet eine Formel, mit der die Masse für eine Menge C ($C \neq \emptyset$) als Kombination berechnet werden kann:

$$m_{12}(C) = \frac{1}{1-K} \sum_{A\cap B=C} m_1(A)m_2(B)$$

In der Formel für die Kombination der Massenfunktionen findet sich der Wert K, der angibt, wie unterschiedlich die Theorien sind, die miteinander kombiniert werden sollen, und der für die Normalisierung genutzt wird, damit die Summe der Massen in der kombinierten Funktion 1 bleibt:

$$K = \sum_{A\cap B=\emptyset} m_1(A)m_2(B)$$

Zusammenfassend bietet die D-S Evidenztheorie eine Möglichkeit, verschiedene Sensorquellen in Anbetracht ihrer Unsicherheiten zu fusionieren. Die Massenfunktionen, welche die besten Ergebnisse erzielen, können empirisch ermittelt werden. Der Bereich der zu testenden Funktionen kann über mehrere Schritte eingeschränkt werden. Allgemein kann eine Massenfunktion konstruiert werden, indem dem Ergebnis eines Anomalieerkennungssystems ein Wert für die Gewissheit zugewiesen wird. Wird also davon ausgegangen, dass Anomalien mit einer 80-prozentigen Sicherheit bestimmt werden können, gilt für die Menge $A = \{Anomalie\}$ die Masse $m(A) = 0,8$. In der Potenzmenge $P(Z)$

von $Z = \{Anomalie, Normal\}$ bleiben neben der leeren Menge noch $N = \{Normal\}$ und Z selbst übrig. Daher kann die verbleibende Masse von $1 - 0,8 = 0,2$ auf N und Z verteilt werden. Da die Massen aller Funktionen bei der Fusion multipliziert werden, ist es wenig sinnvoll, $m(\{Normal\})$ auf 0 zu setzen. Weiterhin muss durch die Fusion zwangsläufig eine Entscheidung getroffen werden, weshalb die Gesamtmasse von 1 bei der Konstruktion der Massenfunktion immer ausschließlich auf die disjunkten Mengen A und N verteilt wird. In dem Beispiel ist $m(A) = 0.2$, sodass gilt $\sum_{M|M\in P(Z)} m(M) = 1$. Auch in der Gegenrichtung kann eine Massenfunktion konstruiert werden, indem ein Wert für die Gewissheit für normale Ergebnisse festgelegt wird. Die Gewissheiten für Anomalien und normale Klassifikationen müssen dabei nicht identisch sein.

Der schlechtestmögliche Wert für die Gewissheit eines Anomaliedetektors liegt bei 50%, da dies einer Massenfunktion entspricht, welche aussagt, dass ein beliebiges Intervall mit gleicher Wahrscheinlichkeit entweder als Anomalie oder als Normal klassifiziert wird. In diesem Fall liefert der Detektor keine verwertbare Information, da seine Entscheidungen rein zufällig sind und keine echte Trennung zwischen Anomalien und normalen Dateninstanzen ermöglicht wird. Durch diesen Umstand ergibt sich eine sinnvolle Eingrenzung der Massen auf Werte größer als 0,5, da niedrigere Werte keine verwertbare Information liefern.

Fusion über die Zeit

Eine weitere Möglichkeit für die Datenfusion ist die Fusion über die Zeit. Grundlegend steht hier der Gedanke dahinter, dass die Auslöser für Anomalien für einen gewissen Zeitraum andauern und somit Anomalien auslösen. Dies hat zur Folge, dass es nahezu keine Punktanomalien in Form einzelner als Anomalie klassifizierter Datenpunkte geben kann. Dies hat zur Folge, dass Anomalien nur in Gruppen auftreten. Ein beliebtes Mittel für die Entfernung einzelner Ausreißer ist ein Median-Filter. Hierbei werden die Datenpunkte im Bereich des Filters sortiert und der in der Mitte liegende Wert wird als Ergebnis gewählt. Bei einer binären Klassifikation hat das zur Folge, dass das im Fenster häufiger vorkommende Ergebnis als Medianwert gewählt wird. Mittels des Sliding-Window-Verfahrens kann ein Fenster mit fester Größe über die klassifizierten Dateninstanzen geschoben werden, sodass das Ergebnis über die Zeit geglättet werden kann.

Da es sich um eine binäre Klassifikation handelt, kann außerdem ein Durchschnittsfilter verwendet werden, um die Punktanomalien zu entfernen, denn auch dieser Filter ermittelt den häufiger vorkommenden Wert innerhalb des Fensters. Die Größe des Fensters ist dabei in beiden Fällen von entscheidender Bedeutung, da durch sie eine minimale Dauer von Anomalien vorgegeben wird, die erkannt werden kann. Liegt die Größe des Fensters bei 11 Sekunden, können bei einer Intervallgröße von einer Sekunde und perfekter Klassifizierung keine Anomalien erkannt werden, die weniger als 6 Sekunden andauern. Das Fenster darf auf der anderen Seite nicht zu klein sein, da der Filter sonst nicht funktionieren kann. Die Größe des Intervalls ist proportional zur Fenstergröße, sodass das Fenster bei einer Intervallgröße von einer Sekunde mindestens 3 Sekunden umfassen muss.

Kombinationslösungen

Die Datenfusion ist kein Schritt, der nur einmal durchgeführt werden könnte. Daher bieten sich Kombinationslösungen an, um die verschiedenen Stärken der einzelnen Fusionsmöglichkeiten nutzen zu können. So bietet die Fusion bspw. unabhängig von anderen Verfahren die Möglichkeit, einzelne Ausreißer im zeitlichen Kontext entfernen zu können, mit dem Hintergedanken, dass der Auslöser für Anomalien eine gewisse Zeitspanne aktiv bleibt und die erzeugten Anomalien somit nicht allein auftauchen.

Auch die einzelnen Verfahren können mehrfach eingesetzt werden, da nicht strikt definiert ist, wie die Datenfusion konkret definiert ist. So könnte ein erster Fusionsschritt die Ergebnisse der verschiedenen Algorithmen zusammenfassen und anschließend könnten diejenigen Datenströme, die auf verschiedenen Switches aufgezeichnet werden, zwischen den Switches fusioniert werden.

7.2 Konzeptionelle Überlegungen

Der Zweck der Datenfusion ist die Verbesserung der Gesamterkennungsleistung und Präzision im Vergleich zu den Ergebnissen der einzelnen Instanzen. Dafür muss zunächst geklärt werden, welche Daten für die Fusion zur Verfügung stehen. Dabei stellt sich zunächst die Frage, welcher Nutzen aus der Fusion gezogen werden kann. Allgemein lassen sich alle implementierten Fusionsverfahren als kompetitiv bezeichnen, da es nicht darum

geht, neue Informationen zu synthetisieren oder das Gesamtbild zu vergrößern, sondern darum, verschiedene Klassifikationsergebnisse zu einem Gesamtergebnis zusammenzuführen, um Fehlklassifikationen zu vermeiden. Das übergeordnete Ziel ist demnach die Verbesserung der Certainty.

7.2.1 Zusammensetzung der Fusionsdaten

Die Zusammensetzung der Daten hängt von den folgenden Parametern bzw. Konfigurationen ab, welche aus dem Kontext der Anomalieerkennung stammen: Anomalieszenario, Anomaliewahrscheinlichkeit im Szenario, Datenstrom, Switch im Fahrzeugnetzwerk, Intervallgröße für die Berechnung der Netzwerkmetriken, Submenge der möglichen Netzwerkmetriken und der Anomalieerkennungsalgorithmus. Das konkrete Anomalieszenario und die konkrete Wahrscheinlichkeit für das Auftreten einer Anomalie geben das Szenario vor, in dem Anomalien erkannt werden sollen. Für die Fusion der Datenströme, die getrennt voneinander analysiert werden und aus einem konkret spezifizierten Szenario stammen, bleibt demnach die Potenzmenge aus: Intervallgrößen, Netzwerkmetrik-Submengen und Algorithmen.

7.2.2 Fusion zwischen Switches

Eine wichtige konzeptionelle Überlegung ist die Vergleichbarkeit der Ergebnisse zwischen verschiedenen Switches. Ist es möglich und sinnvoll, die Ergebnisse der Anomalieerkennung von verschiedenen Switches zu fusionieren? Diese Frage stellt sich, da ein Paket auf seinem Weg durch das Netzwerk in den Aufzeichnungen verschiedener Switches landet. Demnach könnte es sinnvoll sein, nicht nur auf einem Switch zu fusionieren, sondern auch zwischen den Switches. Darüber hinaus sollte die Fusion sicherstellen, dass es für einen Datenstrom nicht mehrere Klassifikationen gleichzeitig gibt. Dies wäre der Fall, wenn ein Datenstrom auf mehreren Switches auftaucht, da die Anomalieerkennung für jeden Switch getrennt arbeitet. Dies würde dazu führen, dass es potentiell widersprüchliche Klassifikationen für einen Datenstrom zu einem Zeitpunkt geben könnte. Dieser Konflikt muss so oder so aufgelöst werden, da bei einem uneindeutigen Ergebnis keine Gegenmaßnahmen eingeleitet werden können. Die Fusion ist für eine solche Auflösung der ideale Zeitpunkt.

Ein wichtiger Aspekt dabei, der auch für die Fusion auf einem Switch relevant ist, ist die Definition des Zeitpunktes und das Feststellen von Gleichzeitigkeit. Die Anomalieerkennung klassifiziert nicht einzelne Pakete, sondern Intervalle von Paketen. Die zeitlichen Grenzen eines Intervalls richten sich dabei nach den Paketen oder Frames innerhalb des Intervalls auf einem Datenstrom. Es muss möglich zu bestimmen sein, ob zwei Intervalle gleichzeitig sind oder nicht, da ansonsten nicht zwischen den Switches fusioniert werden kann. Geht man von einer einzelnen Übertragungseinheit (Paket oder Frame) aus, die durch das Netzwerk übertragen werden soll, dann würde sich diese nicht plötzlich vom Start zum Ziel, sondern im Laufe der Zeit durch das Netzwerk bewegen. Zu einem Zeitpunkt x würde dann ein Intervall für die Berechnung der Netzwerkmetriken auf dem ersten Switch A beginnen und zu Zeitpunkt y auf dem zweiten Switch B. Da abhängig von der Auflösung des Paketzeitstempels der Wert von y größer sein kann als der von x, beginnen die Intervalle auf den Switches A und B zu unterschiedlichen Zeitpunkten, sodass es nicht möglich ist, die Startzeitpunkte der Intervalle auf Identität zu vergleichen. Stattdessen muss ein geeigneter Wert für die maximale Zeitdifferenz gefunden werden, bei dem zwei oder mehr Intervalle als gleichzeitig angesehen werden. Genau das gleiche gilt für den Fall, dass das letzte Paket in einem Intervall den Zeitstempel des Intervalls definiert. Stellt man sich die Zeitstempel der Intervalle als Mittelpunkte von zweidimensionalen Behältern auf einem Zeitstrahl vor, in welche die Intervalle der anderen Switches einsortiert werden sollen, dann berühren sich die Grenzen der Behälter bei einem Toleranzwert von einer halben Intervallgröße in jede Richtung. Bei größeren Werten würden sich die Grenzen überschneiden, sodass eine eindeutige Zuordnung nicht möglich wäre. Bei kleineren Werten würde es Lücken zwischen den Grenzen geben, sodass bei der Fusion neue Intervalle entstehen würden, da es nicht immer eine passende Zuordnung gibt. Ein geeigneter Wert könnte demnach genau bei einer halben Intervallgröße liegen. Auf diese Weise können einem Datenstrom mit regelmäßigen Intervallgrenzen beliebige Intervalle der anderen Switches zugeordnet werden. Es ist also sichergestellt, dass bei der Intervallfusion keine neuen Intervalle entstehen, es aber immer eine eindeutige Zuordnung gibt.

Nachdem geklärt ist, wie die Gleichzeitigkeit von Intervallen bestimmt werden kann, muss festgelegt werden, was passiert, wenn die Klassifizierungen der Intervalle zu einem Zeitpunkt unterschiedlich sind. Dabei stellt sich zunächst die Frage, was ein solcher Fall fachlich bedeutet und wie es dazu kommen kann. Dies kann an einem Beispiel verdeutlicht werden: Es wird ein Datenstrom in einem Netzwerk betrachtet, welches aus mindestens zwei Switches besteht, auf denen der Datenstrom vorkommt. Der Startpunkt des

Datenstroms ist ein Sensor, der seine Daten dem Zielpunkt zusendet. Die Route geht dabei vom Start über Switch A und Switch B bis zum Ziel. Die Anomalieerkennung arbeitet getrennt auf den beiden Switches, sodass zu jedem Zeitpunkt zwei Klassifikationsergebnisse vorliegen. Der Angriff auf das Netzwerk setzt in der Verbindung zwischen Switch A und B an und eliminiert 50% der Pakete. Auf Switch A lässt sich der Angriff nicht erkennen, dafür aber auf Switch B. Hat die Anomalieerkennung korrekt klassifiziert, liegt nun folgendes Ergebnis vor: Switch A: Klassifikation normal; Switch B: Klassifikation Anomalie. In diesem Beispiel liegt tatsächlich ein Angriff vor, weshalb die Fusion zu dem Ergebnis kommen sollte, dass eine Anomalie vorliegt. Wichtig ist, dass bei der Fusion ohne Informationen über die Netzwerktopologie keine Information über die Reihenfolge der Ergebnisse vorliegt. Demnach kann nicht ermittelt werden, ob Switch A oder B der erste im Netzwerk ist. Würde Switch B der erste sein, so wäre es wahrscheinlich, dass es sich um eine Fehlklassifikation handelt. Andernfalls würde dies nämlich bedeuten, dass die Anomalie verschwunden ist und die eliminierten Pakete wieder aufgetaucht wären. Da dies sehr unwahrscheinlich ist, kann nur überlegt werden, wo die Fehlklassifikation aufgetreten ist. Der *Evaluation der Anomalieerkennung* (Kapitel 6) lässt sich entnehmen, dass die meisten Algorithmen in der Lage sind, alle Anomalien zu erkennen und eher ein Problem mit FPs vorliegt. Aus diesem Grund liegt der Schluss nahe, dass es in dem Beispiel keine Anomalie gab.

Das Beispiel soll zeigen, dass nur anhand der Information, dass es zu widersprüchlichen Klassifikationen kam, nicht abgeleitet werden kann, was die korrekte Auflösung für den Konflikt ist. Stattdessen benötigt man weitergehende Informationen über die Topologie des Netzwerks und über die Performance der Algorithmen. Ohne dieses zusätzliche Wissen kann nicht ausgeschlossen werden, dass es sich um den ersten Fall im Beispiel und damit um eine Anomalie handelt. Aus diesem Grund wird in der naiven Implementation der Fusion zwischen den Switches immer davon ausgegangen, dass es sich um eine Anomalie handelt, sobald eine Anomalie-Klassifikation vorliegt, unabhängig von der Anzahl der Gegenstimmen. Eine Implementierung, die solche zusätzlichen Informationen in den Fusionsprozess einbeziehen kann, könnte für zuverlässigere Ergebnisse sorgen, würde jedoch den Rahmen dieser Arbeit überschreiten und wird im weiteren Verlauf nicht berücksichtigt.

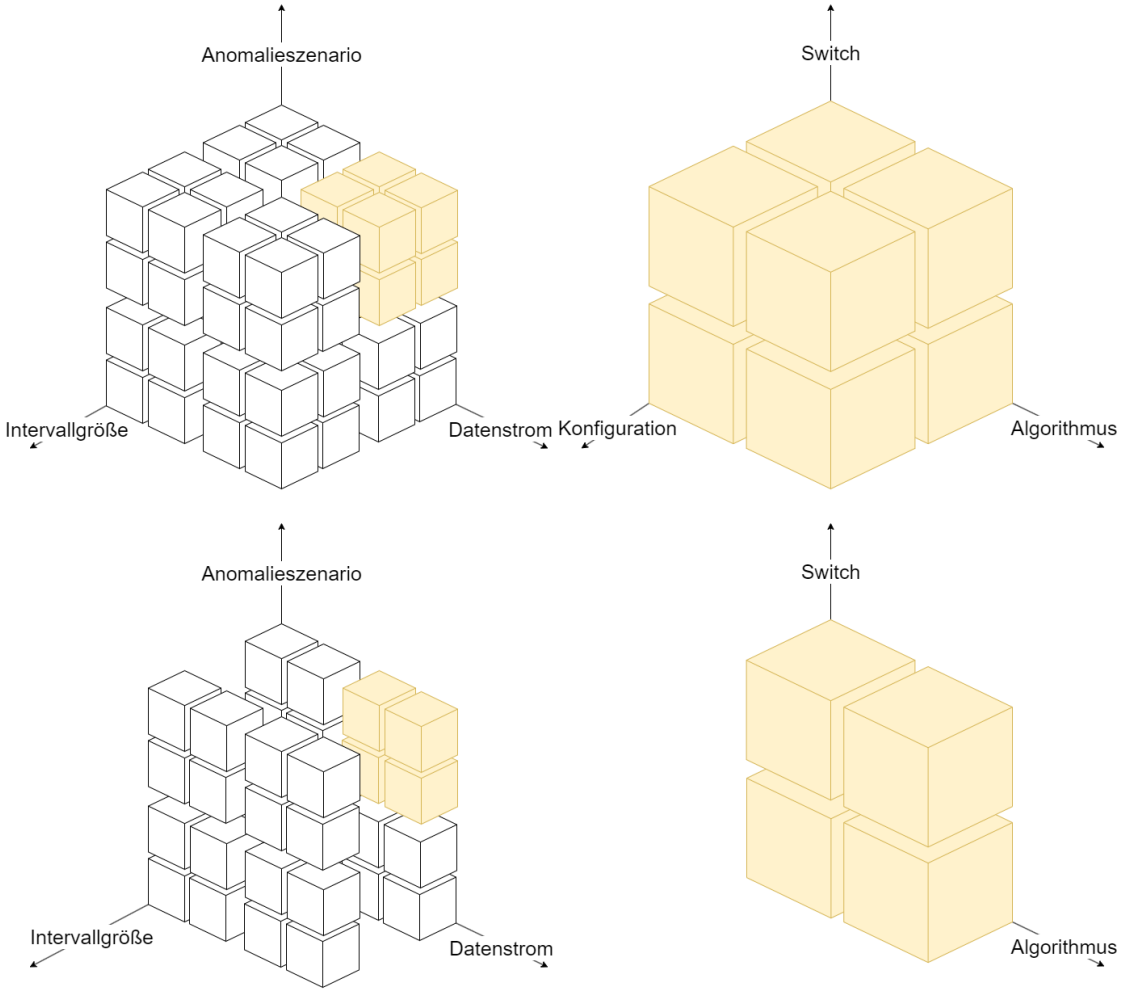


Abbildung 7.1: Visualisierung der Datenfusion Teil eins: Auf dem Diagramm lässt sich der erste Schritt der Datenfusion erkennen, in welchem die Dimension der Konfiguration aufgelöst wird. Oben vor und unten nach dem ersten Fusionsschritt. In den übrigen Schritten der Fusion werden in Abbildung 7.2 dargestellt.

7.2.3 Aufbau der Datenfusion

Die Datenfusion baut auf den Daten verschieden konfigurierter Algorithmen auf. Das Ziel des Datenfusion-Frameworks, das die zuvor vorgestellten Algorithmen implementiert, ist es, eine geeignete Fusionsmethode (oder eine Kombination mehrerer) zu ermitteln, die in den Anomalieszenarien (vgl. 6) möglichst gute Ergebnisse erbringt.

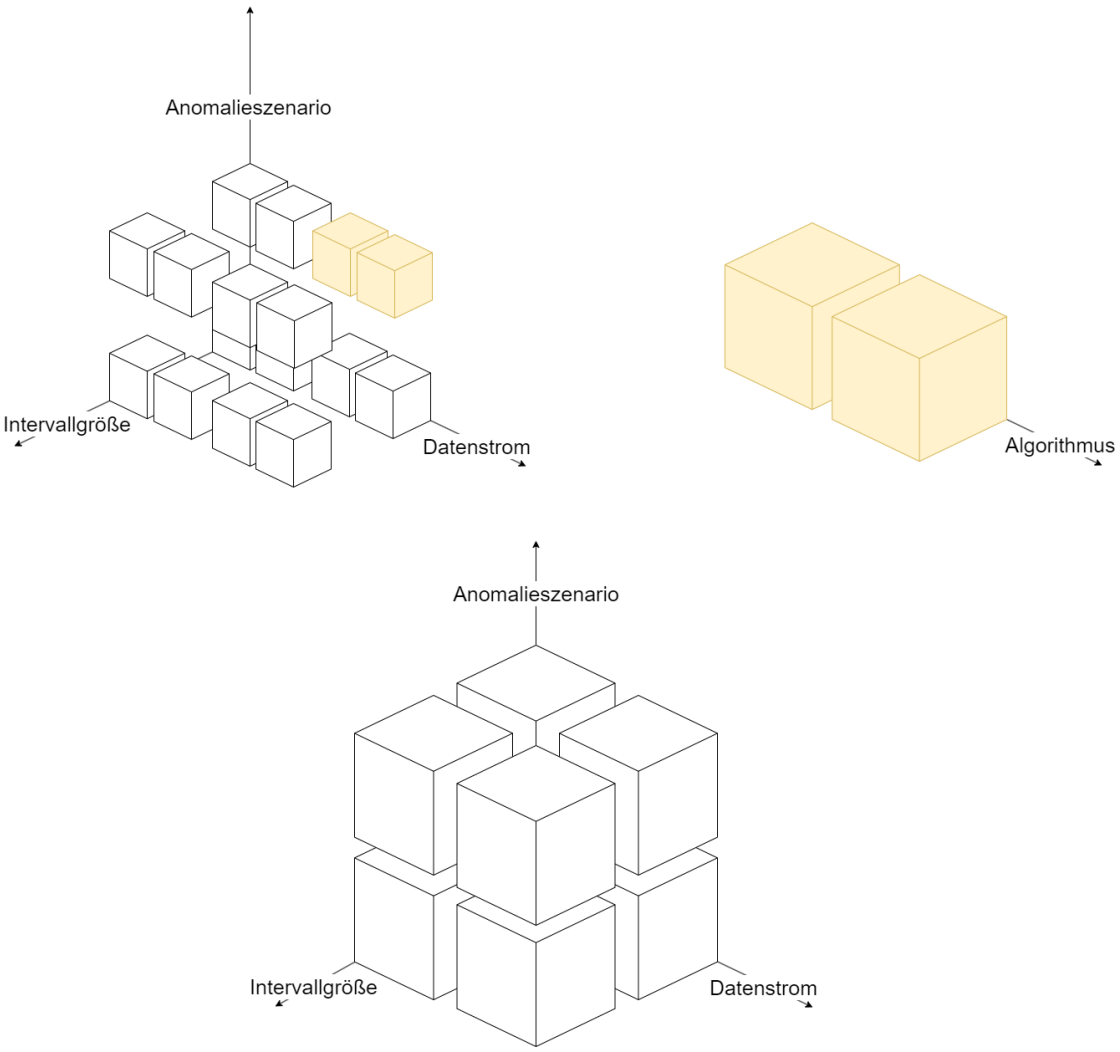


Abbildung 7.2: Visualisierung der Datenfusion Teil zwei: Auf dem Diagramm werden die letzten beiden Schritte der Datenfusion visualisiert. Der Übergang von Abbildung 7.2 zum oberen Teil des Diagramms zeigt die Auflösung der Switch-Dimension. Der Übergang zum unteren Teil zeigt die Auflösung der Algorithmus-Dimension. Übrig bleibt eine Klassifizierung für jede Kombination aus Szenario, Intervallgröße und Datenstrom

Der konzeptionelle Aufbau der Datenfusion ist in den beiden Abbildungen 7.1 und 7.2 dargestellt. Die Abbildung konzentriert sich auf die Visualisierung der verschiedenen Dimensionen, die den Kontext der Fusion vorgeben, und welche Dimensionen durch die Datenfusion kollabiert werden. Die Abbildung wird von oben nach unten gelesen und die gelb markierten Bereiche sollen auf der jeweils rechten Seite vergrößert dargestellt

werden. Im oberen Teil der ersten Abbildung (7.1) lassen sich insgesamt sechs Dimensionen erkennen. Der übergeordnete Würfel, bestehend aus acht Elementen, von denen eines vergrößert dargestellt ist, spannt drei Dimensionen auf: das Anomalieszenario, die Intervallgröße und der Datenstrom. Jeder ‚Punkt' innerhalb des dreidimensionalen Koordinatensystems besteht seinerseits aus drei Dimensionen: den verschiedenen Switches des Fahrzeugnetzwerks (siehe Abbildung 4.2), die als Beobachtungspunkte dienen, den verschiedenen Algorithmen, mit welchen die Anomalieerkennung durchgeführt wurde, und den Konfigurationen der Algorithmen, die sich aus den Parametern der Algorithmen und den verschiedenen Netzwerkmetrik-Submengen (vgl. Sektion *Beobachtungsschwerpunkte im Netzwerk* 4.3) zusammensetzen. Der erste Schritt der Datenfusion kollabiert die Konfigurations-Dimension. Mittels der Mehrheitsfusion oder der D-S-Evidenztheorie werden die verfügbaren Informationsquellen eines Algorithmus zu einem Gesamtergebnis fusioniert. Der zweite obligatorische Schritt, der auch durch die beiden genannten Verfahren implementiert wird, umfasst die Fusion der Daten von den vier Switches. Dieser Schritt wurde in der vorigen Sektion 7.2.2 genauer erklärt. Nachdem durch die Fusion die beiden Dimensionen entfernt wurden, bleibt ein Klassifikationsergebnis für jede der drei übergeordneten Dimensionen übrig. An dieser Stelle setzt die Fusion über die Zeit an, welche diese Ergebnisse glättet, um einzelne Fehlklassifikationen oder kleine Cluster zu glätten. Die Kombination der drei Fusionsverfahren spannt damit vier mögliche Kombinationen auf: Mehrheitsfusion und D-S einmal mit und einmal ohne anschließende Fusion über die Zeit. Die so entstehenden Ergebnisse können dann evaluiert werden und der vielversprechendste Kandidat kann in einem Produktivsystem verwendet oder weiter getestet werden.

Anstatt einen fusionierten Algorithmus auszuwählen, kann außerdem ein weiterer Fusionsschritt hinzugefügt werden. Dabei werden die Ergebnisse aus der ersten Fusion noch ein weiteres Mal fusioniert. Dieser Schritt kann potentiell dabei helfen, die Fehlklassifikationen und insbesondere die FPs weiter zu senken. Dieser zweite Fusionsschritt kann wie auch zuvor durch eines der beiden primären Fusionsverfahren ausgeführt werden. Doch da die Algorithmen sehr unterschiedliche Ergebnisse liefern, ist die Flexibilität des D-S-Verfahrens für diesen zweiten Fusionsschritt von besonderem Interesse und die Mehrheitsfusion wird nicht betrachtet. Im weiteren Verlauf dieser Arbeit wird der Ablauf der Fusion: D-S, Fusion über die Zeit (erste Ebene: Konfigurationen und Switches werden fusioniert) D-S und Fusion über die Zeit (zweite Ebene: Algorithmen werden fusioniert) als MLDS bezeichnet.

8 Umsetzung der Datenfusion

Die Umsetzung der Datenfusion basiert auf den beiden Modulen DataFusion und Evaluator (siehe Abbildung 5.1). Das Evaluator-Modul ist als eine Klasse implementiert und hat die Aufgabe, verschiedene Log-Dateien der einzelnen Anomalieerkennungs-Instanzen zu lesen und das Gesamtergebnis in eine Datei zu schreiben. Die Datenstruktur, in der das Ergebnis gespeichert wird, ist ein Dictionary bzw. eine Map und wird als JSON-Datei gespeichert. Im Dictionary finden sich als Schlüssel-Werte die Dateipfade und in den zugehörigen Daten die Metadaten zur Datei (das Anomalieszenario, der verwendete Algorithmus etc.) sowie eine Liste der Auswertungen mit Zeitstempeln, den ausgewerteten Netzwerkmetriken und den Klassifikationscodes.

Das Modul DataFusion besteht aus mehreren Klassen, wobei eine Basisklasse allgemeine Funktionalitäten implementiert und von dieser Basisklasse erbende Klassen die jeweiligen Fusionsverfahren realisieren. Bei der Erzeugung eines DataFusion-Objekts muss u. a. ein Pfad mitgegeben werden, unter dem die Ergebnisse der Anomalieerkennung zu finden sind. Falls diese Ergebnisse noch nicht vom Evaluator-Modul ausgewertet wurden, arbeitet die Basisklasse mit einem Thread-Pool, um die Auswertung möglichst schnell durchzuführen. Zunächst wird überprüft, in welcher Tiefe sich der gegebene Auswertungs-Pfad befindet, um zu ermitteln, welche Daten für die Fusion zur Verfügung stehen. Dabei kann sich beispielsweise herausstellen, dass alle Daten zu einem spezifisch konfigurierten Anomalieszenario verfügbar sind oder nur die Daten eines Algorithmus und einer spezifischen Intervallgröße innerhalb des Szenarios. Die Ergebnisse des Evaluator-Moduls werden in einem eigenen Verzeichnis gespeichert, um später darauf zugreifen zu können.

Die Fusionsverfahren der erbenden Klassen umfassen einerseits Methoden zur Fusion von Daten zu einem Zeitpunkt aus verschiedenen Quellen und andererseits Verfahren zur Fusion über die Zeit.

Mehrheitsfusion

Das erste und simpelste implementierte Verfahren ist die Mehrheitsfusion, die bereits im *Konzept-Kapitel* erläutert wurde. Dabei wird eine Submenge aller verfügbaren Daten für ein spezifisches Szenario zu einem spezifischen Zeitpunkt durch ein spezielles Suchverfahren erzeugt. Die zusammengehörigen Datenpunkte werden anhand von Attributen wie Anomalieszenario, Datenstrom, Intervallgröße etc. ermittelt. Nach der Zusammenstellung der Datenquellen wird die prozentuale Verteilung der Anomalie- und Nicht-Anomalie-Klassifikationen berechnet. Liegt der Anteil der Anomalien unter einem Grenzwert, wird das Gesamtergebnis als normal klassifiziert; Liegt er darüber, wird eine Anomalie erkannt. Zusätzlich zum Ergebnis wird für die Evaluation außerdem die Information gespeichert, ob es sich bei dem fusionierten Intervall tatsächlich um eine Anomalie gehandelt hat oder nicht, was auch für die anderen Verfahren gilt. Die eigentliche Fusion erfolgt in zwei Phasen, sodass zuerst alle Daten innerhalb eines Switches fusioniert werden und erst danach die Ergebnisse zwischen den Switches. Dies ist nötig, da die Angriffe nur einen Teil des Netzwerks betreffen können und es dadurch passieren kann, dass ein Datenstrom zwar auf allen Switches aufgezeichnet wurde, Anomalien aber nur auf einem Switch erkennbar sind. Diese Möglichkeit limitiert die Optionen für eine Fusion auf Switch-Ebene. Deshalb klassifiziert die Fusion ein Intervall als Anomalie, sobald nur auf einem Switch Anomalien erkannt wurden. Dies sorgt dafür, dass Angriffe, die nur einen Switch betreffen, erkannt werden können.

Dempster-Shafer-Fusion

Ein weiteres implementiertes Verfahren zur Datenfusion ist die D-S-Evidenztheorie, mit welcher die Wahrscheinlichkeits- oder Masseverteilungen verschiedener Quellen fusioniert werden können. Wie bei der Mehrheitsfusion werden auch für die Ausführung der D-S-Implementation passende Datenquellen zusammengetragen und anschließend Intervall für Intervall fusioniert. Auch hier ist die Fusion zweigeteilt. Zunächst werden die Ergebnisse innerhalb eines Switches fusioniert, um anschließend das Ergebnis über alle Switches zu bestimmen. Dabei gelten dieselben Überlegungen wie bei der Mehrheitsfusion und ein Intervall wird als Anomalie klassifiziert, sobald auf einem Switch Anomalien erkannt wurden.

Die Fusion kann auch mehrfach durchgeführt werden, um neben der Fusion von Algorithmus-Konfigurationen und Switches auch die Ergebnisse verschiedener Algorithmen zu fusionieren (MLDS-Verfahren; vgl. Sektion 7.2.3).

Ein für das D-S-Fusionsverfahren entscheidender Punkt ist die Bestimmung der Massenfunktionen. Wie in der Sektion 7.1 erklärt, können die Massenfunktionen konstruiert werden, indem ein Wert für die Anomalie- und einer für die Normalmasse festgelegt wird. Um optimale Ergebnisse zu erhalten, ist die Brute-Force-Methode (also das Ausprobieren aller möglichen Werte) eine sichere Wahl. Oft besteht das Problem, dass es zu viele Möglichkeiten gibt und der Rechenaufwand zu groß wird. Konkret auf die primäre Fusion bezogen gibt es bei x Algorithmen $x \cdot y \cdot z$ Kombinationen, wobei y und z für die mögliche Anzahl von Massen für Anomalie- und normale Klassifizierungen stehen. Entscheidet man sich bei sechs Algorithmen für jeweils zehn Massen, ergibt das $6 \cdot 10 \cdot 10 = 600$ Kombinationen, was definitiv im Bereich des Möglichen liegen sollte.

Anders verhält sich das bei der sekundären Fusion. Hier skalieren die Kombinationen exponentiell mit der Zahl der Algorithmen, da die Massenfunktionen der Algorithmen nicht mehr getrennt voneinander betrachtet werden können: $(y \cdot z)^x$. Wählt man also erneut sechs Algorithmen bei jeweils zehn Massen, ergibt das $(10 \cdot 10)^6 = 100^6 = 10^{12}$, also eine Billion Kombinationen, was die Brute-Force-Methode für die zweite Stufe sehr ineffizient macht.

Einen guten Anhaltspunkt für die Berechnung von Massenfunktionen können die Ergebnisse der Algorithmen liefern. Dies ist möglich, da bspw. ein durchschnittlicher Recall von 1 verrät, dass 100% der Anomalien erkannt wurden, was einer Masse von 1 entsprechen könnte. Ein Recall von 0 auf der anderen Seite würde einer Masse von 0,5 entsprechen, da dieser Wert eine maximale Unsicherheit ausdrückt (vgl. Sektion 7.1). Mit dieser Einschätzung muss eine geeignete Abbildung gefunden werden, die den Recall bzw. die Präzision zwischen 0 und 1 auf den Bereich der Masse zwischen 0,5 und 1 abbildet. Eine geeignete Funktion lässt sich in Abbildung 8.1 erkennen. Die parametrisierbare Funktion kann konfiguriert werden, um besonders niedrigen oder hohen x-Werten mehr oder weniger Gewicht zu verleihen. Ein Wert von a nahe 0 entspräche dabei einer linearen Abbildung. Damit bietet die Funktion eine anpassbare Möglichkeit, Evaluationsmetriken wie Recall oder Präzision in Massenwerte zu überführen, ohne einen zu hohen Rechenaufwand zu verursachen.

Da für den zweiten Durchlauf in der Fusion mehrere mögliche Quellen aus der ersten Stufe zur Verfügung stehen, muss entschieden werden, welche davon genommen wird. In

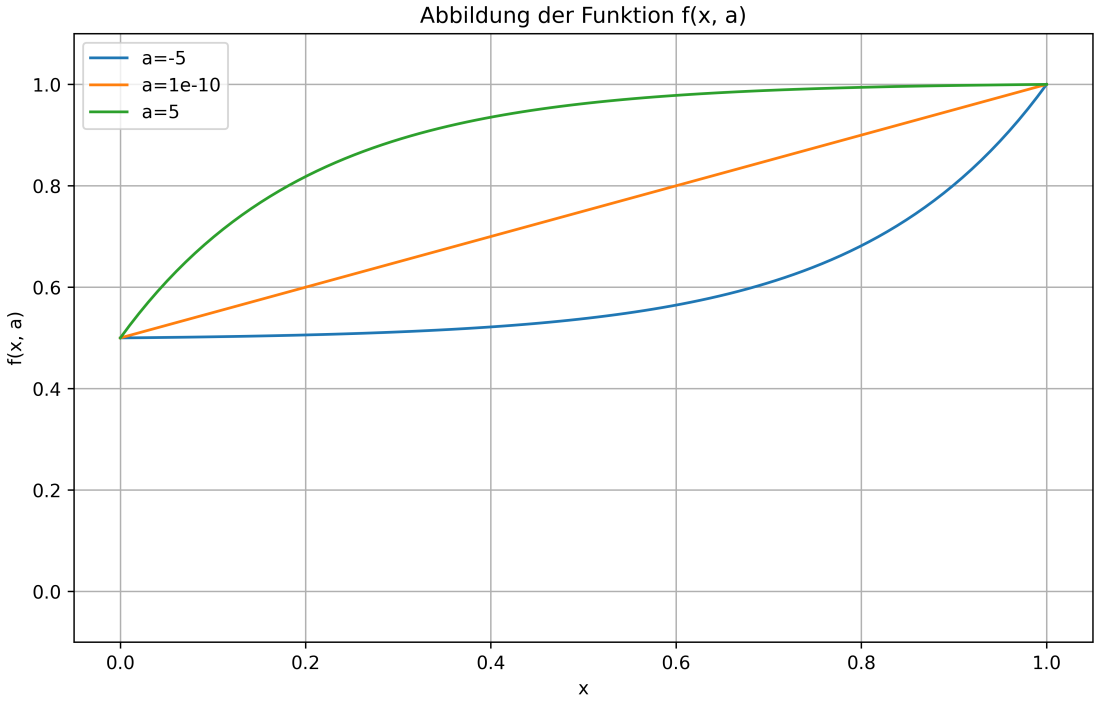


Abbildung 8.1: Visualisierung der parametrisierbaren Funktion $f(x, a) = 0.5+0.5 \cdot \frac{1-e^{-ax}}{1-e^{-a}}$ für die Abbildung von Recall/Präzision auf die Masse für die D-S-Fusion.

der ersten Stufe wurden verschiedene mögliche Massenfunktionen für jeden Algorithmus ausprobiert, die in den Szenarien unterschiedlich gute Ergebnisse gebracht haben. Es ergibt sich von selbst, dass bessere Ergebnisse aus der ersten Stufe für bessere Ergebnisse in der zweiten Stufe sorgen werden. Da das Ziel der Fusion ist, Parameter zu ermitteln, die insgesamt für gute Ergebnisse sorgen, müssen in allen Szenarien dieselben Parameter in der ersten Stufe verwendet werden. Daher muss ein Maß für die Leistung gewählt werden, dessen Durchschnittswert über alle Szenarien hinweg am besten ist. Hierfür bietet sich der F1-Score an, da er den Recall und die Präzision zu einer Metrik kombiniert.

Um diese Überlegung umzusetzen, evaluiert die Implementation automatisch die Ergebnisse aus der ersten Stufe in allen zur Verfügung stehenden Szenarien. Anschließend wird der allgemein beste Kandidat für jeden Algorithmus ermittelt und als Grundlage für die zweite Fusionsstufe genommen.

Insgesamt sollte dieses Verfahren für eine stabile Grundlage sorgen und könnte auch für den Einsatz in einem Produktivsystem eingesetzt werden, indem auch dort Testszenarien durchgeführt und die Leistung gemessen wird. Anhand dieser Testläufe könnte man

mit derselben Herangehensweise passende Massenfunktionen für die erste Fusionsstufe ermitteln.

Fusion über die Zeit

Zuletzt besteht die Möglichkeit zur Fusion über die Zeit. Hierbei kann eine Fenstergröße angegeben werden, die darüber entscheidet, wie viele der vorangegangenen und nachfolgenden Ergebnisse herangezogen werden. Dieses Fusionsverfahren ist nicht dafür gedacht, für sich allein genutzt zu werden. Die Implementation erwartet stattdessen die Ergebnisse eines der anderen Fusionsverfahren als Input-Daten. Der Grund dafür liegt darin, dass es andernfalls nur eine zugrundeliegende Informationsquelle gibt. Stehen zehn potentielle Quellen zur Verfügung, würde es mit der Fusion über die Zeit gleichermaßen zehn Ergebnisse geben, die anschließend fusioniert werden müssten. Da das Verfahren demnach nicht allein eingesetzt werden kann oder sollte, wurde die Fusion über die Zeit als sekundäre Fusion implementiert.

Alle Fusionsverfahren lassen sich ohne Probleme parallelisieren. Hierfür wird ein Worker-Pool eingesetzt, mit dessen Hilfe auf der Ebene der Datenströme parallelisiert wird. Es existiert also eine Task pro Datenstrom in der Netzwerkaufzeichnung, die unabhängig von den anderen Datenströmen ausgeführt werden kann.

Nachdem die Datenfusion durchgeführt wurde, müssen die Ergebnisse evaluiert werden. Die korrekte Evaluation der Fusionsergebnisse erfolgt durch die Analyse der Klassifikationscodes. Falls mindestens ein TP- oder FN-Code vorhanden ist, liegt tatsächlich eine Anomalie vor. Diese Information wird zusammen mit dem Fusionsergebnis gespeichert, um die Fusion retrospektiv evaluieren zu können.

9 Evaluation der Datenfusion

In diesem Kapitel sollen die Ergebnisse der Datenfusion evaluiert werden. Insgesamt gibt es drei implementierte Verfahren, die zu fünf Fusionsverfahren kombiniert wurden. Zunächst wurden die Mehrheitsfusion und die D-S-Fusion einzeln durchgeführt. Außerdem wurden die Ergebnisse der Verfahren über die Zeit fusioniert. Diese vier Verfahren liefern jeweils ein Ergebnis pro Algorithmus (siehe Abbildungen 7.1 und 7.2). Das letzte Verfahren fusioniert die Ergebnisse der D-S-Fusionen mit anschließender Fusion über die Zeit zu einem Gesamtergebnis.

Die Ergebnisse aller fünf Verfahren beziehen sich jeweils auf die fünf Szenarien aus der Anomalieerkennung (vgl. Kapitel 6). Die vollständigen Tabellen mit den Ergebnissen finden sich in Anhang A.3. Die Algorithmen SVM und AE sind nicht in den Tabellen vertreten, da ihre Ergebnisse in der Anomalieerkennung zu schlecht waren und sie somit nicht für die Datenfusion geeignet sind.

Als Vergleichswert für die Verfahren soll der beste durchschnittliche F1-Score aus der Anomalieerkennung dienen. Dieser Wert des MS-Algorithmus lag bei ca. 51%.

9.1 Mehrheitsfusion

Die Ergebnisse der Mehrheitsfusion sind generell nicht sehr überzeugend. Eine Zusammenfassung der Ergebnisse wird in Abbildung 9.1 visualisiert. Dort lässt sich der durchschnittliche F1-Score der Algorithmen erkennen. Dargestellt sind jeweils die beste und die schlechteste Konfiguration, was sich in diesem Fall auf die Intervallgröße bezieht. Auffällig ist hierbei, dass die Algorithmen HBO und MS am besten abgeschnitten haben, während der IF-Algorithmus sehr schlechte Ergebnisse geliefert hat, da der Algorithmus nahezu keine Anomalie erkannt hat.

Die Tabellen mit den vollständigen Ergebnissen finden sich in der Sektion A.3.1. Dort fällt auf, dass im RC-Szenario so gut wie keine Anomalien erkannt wurden, was eine

Verschlechterung gegenüber der Anomalieerkennung darstellt. Verglichen mit dem MS-Algorithmus aus der Anomalieerkennung lag der beste durchschnittliche F1-Score mit einem Wert von ca. 44% und damit sieben Prozentpunkte unter der reinen Anomalieerkennung.

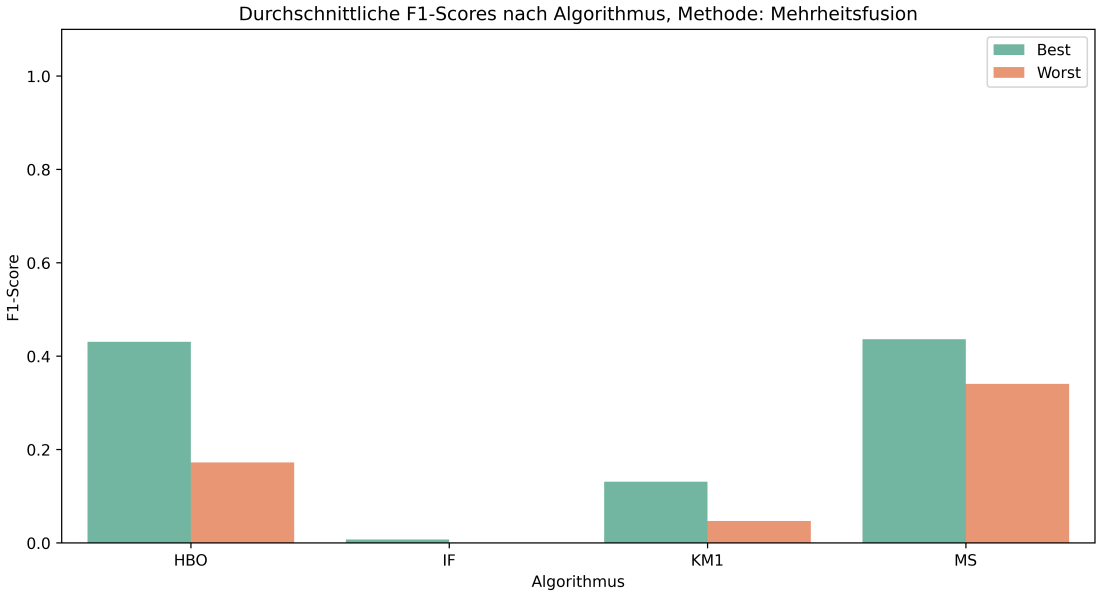


Abbildung 9.1: Die Abbildung zeigt die durchschnittlichen F1-Scores der Algorithmen, fusioniert mit der Mehrheitsfusion. Dargestellt sind jeweils die beste und die schlechteste Konfiguration

9.2 Mehrheitsfusion mit Fusion über die Zeit

Die Ergebnisse der Mehrheitsfusion mit anschließender Fusion über die Zeit sind relativ ähnlich zum vorangegangenen Szenario. Wie sich in der Abbildung 9.2 erkennen lässt, ist der durchschnittliche F1-Score des IF-Algorithmus bei 0 und damit minimal schlechter als im vorigen Szenario. Sichtlich verbessert hat sich der KM-Algorithmus, dessen bester durchschnittlicher F1-Score fast verdoppelt werden konnte. Die vollständigen Ergebnisse finden sich in Sektion A.3.2. Erwähnenswert ist dort der MS-Algorithmus, der es mit der richtigen Konfiguration geschafft hat, die FPR auf 0 zu senken. Verglichen mit dem MS-Algorithmus aus der Anomalieerkennung lag der beste durchschnittliche F1-Score mit einem Wert von ca. 44% und damit sieben Prozentpunkte unter der reinen Anomalieerkennung.

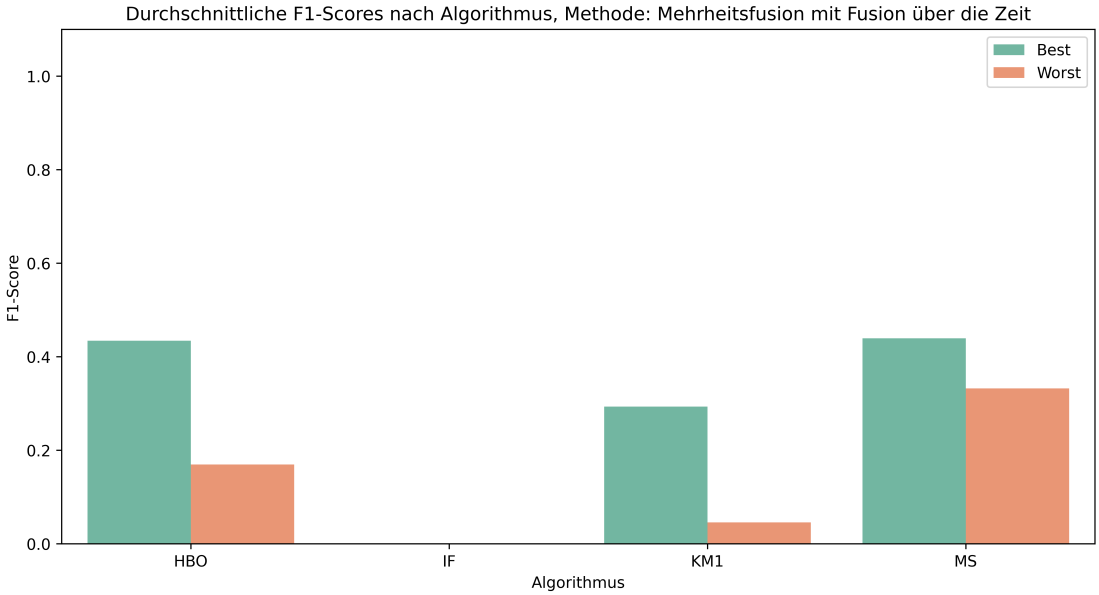


Abbildung 9.2: Die Abbildung zeigt die durchschnittlichen F1-Scores der Algorithmen, fusioniert mit der Mehrheitsfusion mit Fusion über die Zeit. Dargestellt sind jeweils die beste und die schlechteste Konfiguration.

9.3 Dempster-Shafer

In Abbildung 9.3 lassen sich die Ergebnisse der D-S-Fusion erkennen. Obwohl die besten Ergebnisse der Algorithmen recht ähnlich zu den vorigen Methoden waren, fallen doch einige Punkte auf. Der IF-Algorithmus hat es geschafft, einen durchschnittlichen F1-Score von ca. 25% zu erreichen. Den Tabellen aus Sektion A.3.3 lässt sich entnehmen, dass dieser Anstieg vor allem auf das IL-Szenario zurückzuführen ist. Dort hat der Algorithmus in der Anomalieerkennung gute Ergebnisse erzielen können, in der Mehrheitsfusion aber keine Anomalien erkennen können.

Es fällt außerdem auf, dass die Abstände zwischen der besten und der schlechtesten Konfiguration sehr erheblich sind. Dies lässt sich auf das große Spektrum an Konfigurationen zurückführen, welches durch die Brute-Force-Methode bei der Ermittlung der Massenfunktionen zurückzuführen ist. Dies lässt den Schluss zu, dass die Leistung der D-S-Fusion stark abhängig von der richtigen Massenfunktion ist. Kleine Abweichungen in den Funktionen führen jedoch nicht automatisch zu drastisch unterschiedlichen Leistungen. Dies lässt sich in der Abbildung 9.4 erkennen. Dort dargestellt ist der durchschnittliche F1-Score in Abhängigkeit von der Konfiguration der Massenfunktion. Mit

dieser Konfiguration kann eine Massenfunktion konstruiert werden, indem die Masse der Klassifizierung auf den angegebenen Wert und die Masse der anderen Klassifizierung auf 1 – den Wert gesetzt wird (vgl. Sektion 7.1). Der Abbildung lässt sich entnehmen, dass der Wert für die normale Klassifizierung entschiedener ist als der Wert für Anomalien. Der beste Wert dafür scheint der 1 zu sein, was sich auch mit den guten Recall-Werten deckt. Interpretieren könnte man dies so: Ein Recall-Wert von 1 bedeutet, dass alle Anomalien identifiziert werden konnten. Man kann sich also sicher sein, dass ein Intervall normal ist, wenn es als solches klassifiziert wurde. Da die Massenfunktion eine Quantifizierung von (Un-)Sicherheit ist, ergibt die Korrelation zwischen der Masse und der Evaluationsmetrik Sinn.

Auffällig ist die Leistung der Algorithmen im RC-Szenario. Anders als mit den vorigen Verfahren konnten die HBO-, KM- und MS-Algorithmen mit der D-S-Fusion Anomalien erkennen. Verglichen mit dem MS-Algorithmus aus der Anomalieerkennung lag der beste durchschnittliche F1-Score mit einem Wert von ca. 45% und damit sechs Prozentpunkte unter der reinen Anomalieerkennung.

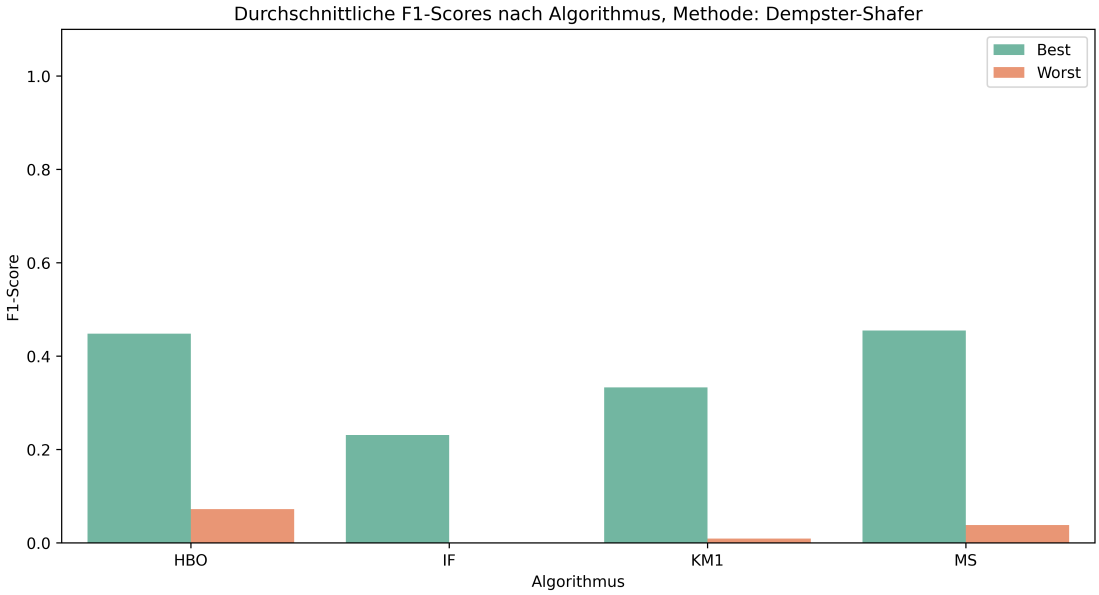


Abbildung 9.3: Die Abbildung zeigt die durchschnittlichen F1-Scores der Algorithmen, fusioniert mit der D-S-Fusion. Dargestellt sind jeweils die beste und die schlechteste Konfiguration.

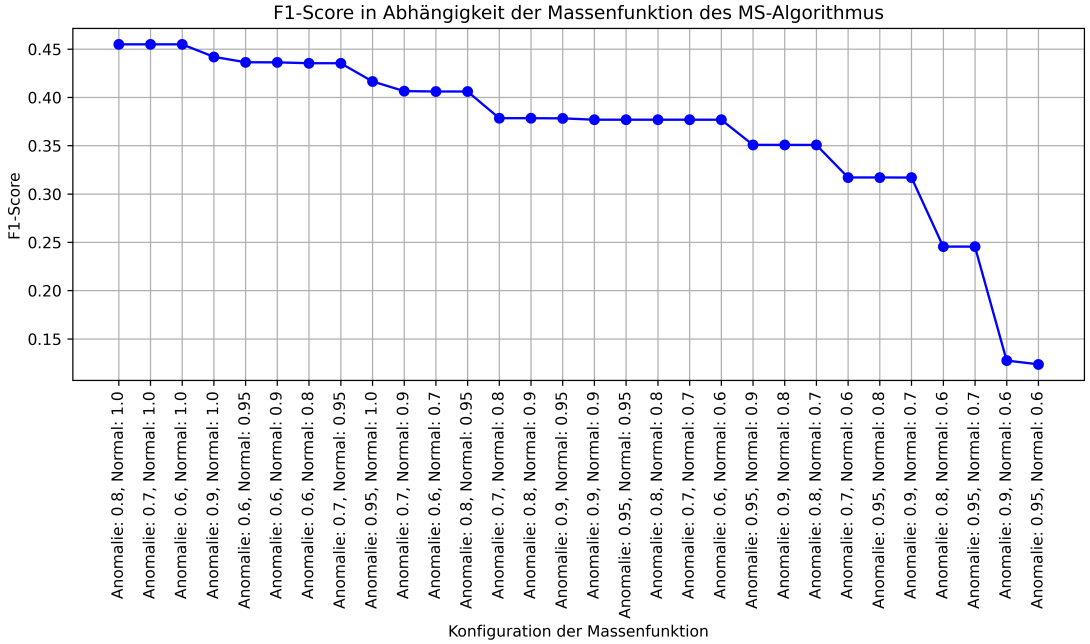


Abbildung 9.4: Die Abbildung zeigt die durchschnittlichen F1-Scores des MS-Algorithmus mit der D-S-Fusion in Abhängigkeit von der Konfiguration der Massenfunktion.

9.4 Dempster-Shafer mit Fusion über die Zeit

Die Ergebnisse aus Abbildung 9.6 lassen erkennen, dass sich die Algorithmen HBO, IF und MS kaum verändert haben. Dies spricht dafür, dass wenig Ausreißer in den fusionierten Daten waren und sich die Ergebnisse somit kaum verändert haben. Anders verhält sich dies mit dem KM1-Algorithmus, dessen F1-Score sich in der besten Konfiguration mit einer Steigerung von ca. 10%, auf der Ebene der beiden besten Algorithmen befindet. Insgesamt scheint die Fusion über die Zeit den größten Einfluss auf den KM-Algorithmus zu haben, da dies auch für die Mehrheitsfusion gültig ist.

Die vollständigen Ergebnisse sind tabellarisch in Sektion A.3.4 aufgeführt. Dort lässt sich erkennen, dass sich die Ergebnisse kaum verbessert haben. Nur die FPR des IF-Algorithmus konnte einen Wert von 0 erreichen. Interessant ist außerdem, dass der HBO-Algorithmus im RC-Szenario nur mit einer Intervallgröße von einer Sekunde Anomalien erkennen konnte. Diese Größe hat sonst i. d. R. für die schlechtesten Ergebnisse gesorgt.

Verglichen mit dem MS-Algorithmus aus der Anomalieerkennung lag der beste durchschnittliche F1-Score mit einem Wert von ca. 48% und damit drei Prozentpunkte unter der reinen Anomalieerkennung.

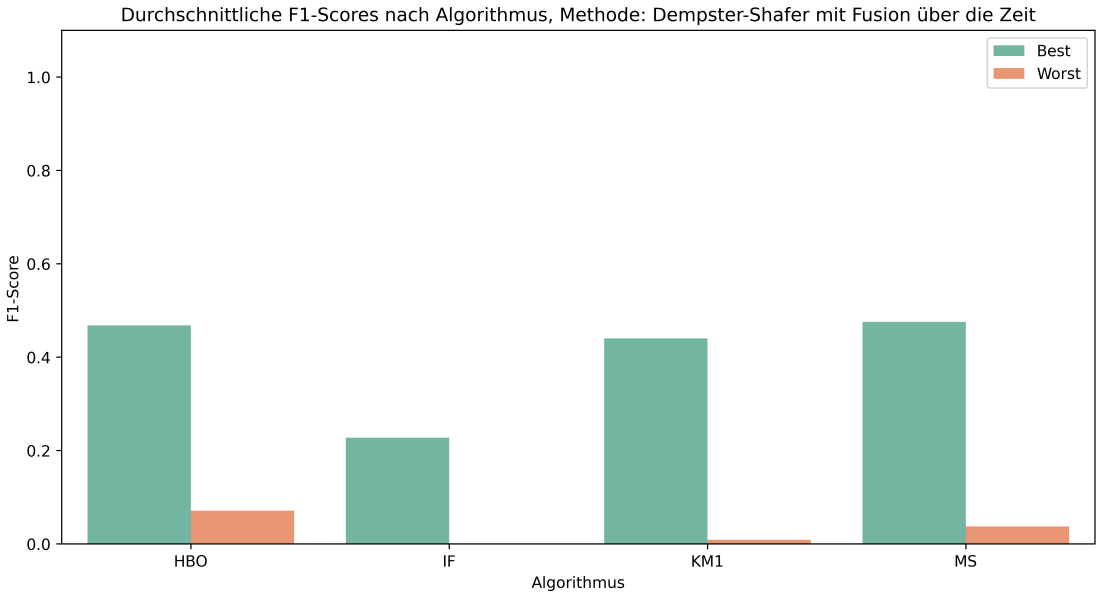


Abbildung 9.5: Die Abbildung zeigt die durchschnittlichen F1-Scores der Algorithmen, fusioniert mit der D-S-Fusion mit Fusion über die Zeit. Dargestellt sind jeweils die beste und die schlechteste Konfiguration.

9.5 Multi-Level Dempster-Shafer

Die MLDS-Fusion ist die einzige in dieser Arbeit behandelte Fusion, die anstatt eines Ergebnisses pro Algorithmus die Ergebnisse aller Algorithmen fusioniert. Somit sind in der Abbildung 9.6 nur zwei Balken zu erkennen. Der Unterschied zwischen der besten Konfiguration (grüner Balken) und der schlechtesten Konfiguration (roter Balken) ist sehr gering. Dies lässt sich auch den Tabellen mit den vollständigen Ergebnissen in Sektion A.3.5 entnehmen. Dort lässt sich erkennen, dass der a-Wert, welcher für die Erstellung der Massenfunktionen relevant ist (vgl. Sektion 8 und Abbildung 8.1), einen sehr geringen Einfluss auf die Leistung der Fusion hat. Nur in wenigen Fällen hat es überhaupt Unterschiede gegeben, sodass in vielen Szenarien identische Ergebnisse für die verschiedenen Werte von a vorliegen.

Insgesamt ist das Ergebnis der MLDS-Fusion besser als alle Einzelergebnisse der Algorithmen mit den anderen Fusionsverfahren. Es scheint also eine stabile Möglichkeit zu sein, die Ergebnisse der Algorithmen ein weiteres Mal zu fusionieren. Besonders überzeugen konnte die Fusion im EAB- und Baseline-Szenario, in denen mitunter perfekte Werte erzielt worden sind. Auf der anderen Seite des Spektrums steht das RC-Szenario, in welchem gar keine Anomalien erkannt werden konnten. Insgesamt hat auch dieses Verfahren noch ein FP-Problem, da diese falschen Klassifizierungen nicht vollständig entfernt werden konnten. Verglichen mit dem MS-Algorithmus aus der Anomalieerkennung lag der beste durchschnittliche F1-Score mit einem Wert von ca. 49% und damit zwei Prozentpunkte unter der reinen Anomalieerkennung.

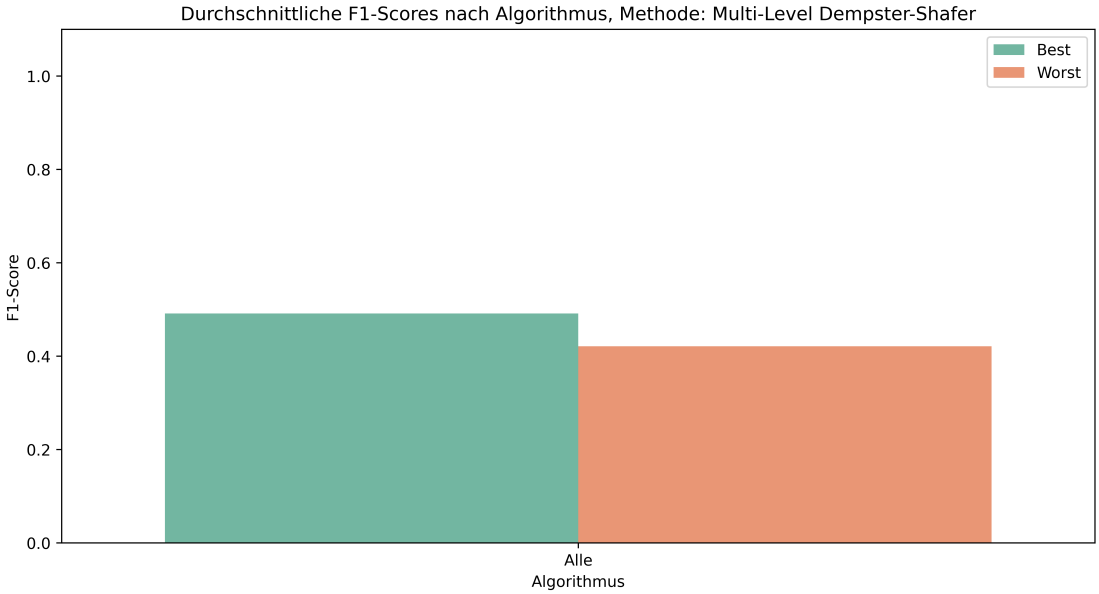


Abbildung 9.6: Die Abbildung zeigt die durchschnittlichen F1-Scores der MLDS-Fusion. Dargestellt sind die beste und die schlechteste Konfiguration.

9.6 Kontextabhängige Bedeutung von False Positives

in FP ist im Kontext dieser Arbeit nicht immer als schlecht anzusehen. So kommt es auf die Position des FP im zeitlichen Kontext an und in welchem Datenstrom es sich befindet. Um dies zu verdeutlichen, eignen sich zwei Beispiele:

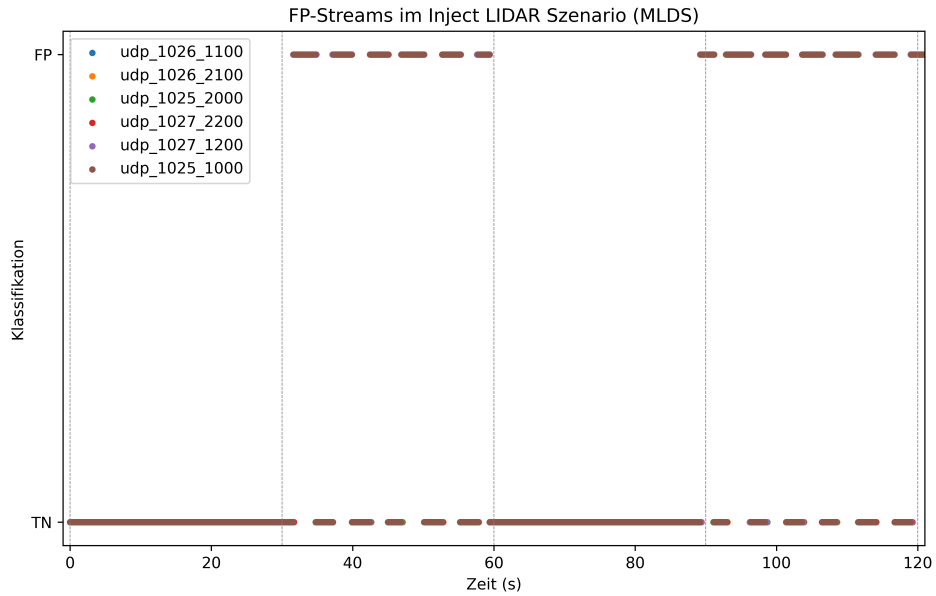


Abbildung 9.7: Die Abbildung zeigt alle Streams im Inject-LIDAR-Szenario, in denen FPs aufgetreten sind. Das Szenario wurde mit dem MLDS-Verfahren fusioniert. Die FPs korrelieren zeitlich mit den tatsächlichen Anomalien.

Zunächst kann es vorkommen, dass ein FP entsteht, der durch die zeitliche Fusion nicht beseitigt wurde, da er unmittelbar an ein tatsächliches Anomalie-Cluster angrenzt. In diesem Fall konnte der genaue Zeitraum der Anomalie nicht präzise bestimmt werden. Dieses Szenario unterscheidet sich grundlegend von dem Fall, in dem eine Gruppe von FPs unabhängig von einer tatsächlichen Anomalie auftritt und dadurch eine nicht notwendige Gegenmaßnahme ausgelöst werden würde.

Ein weiteres Beispiel, dessen Auftreten in den Ergebnissen auch belegt werden kann, bezieht sich auf die Korrelation von FPs in einem Datenstrom mit dem Auftreten von TPs in einem anderen. Dies kann bspw. im IL-Szenario passieren, wenn das Einschleusen der Pakete dazu führt, dass die Weiterleitung von Paketen eines anderen Datenstroms verzögert wird. In diesem Sinne stellt sich die Frage, ob man überhaupt noch von einem FP sprechen kann, da der Datenstrom zwar nicht direkt manipuliert wurde, aber dafür indirekt durch den Anomalie-Auslöser beeinflusst wurde. Die Konsequenz in einem solchen Szenario könnte sein, dass eine potentielle Gegenmaßnahme weniger zielgerichtet eingesetzt werden könnte, da nicht genau bestimmt werden könnte, welcher Datenstrom

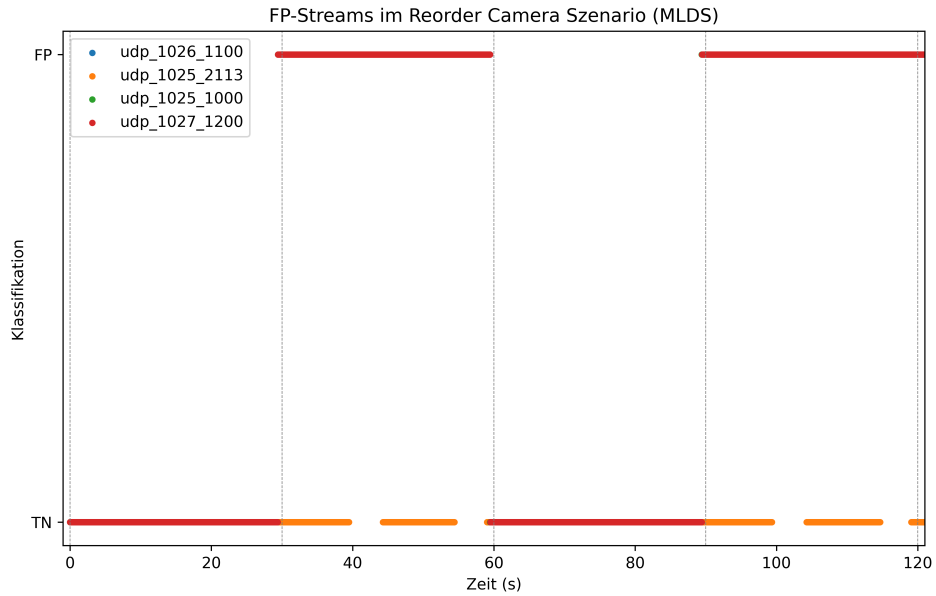


Abbildung 9.8: Die Abbildung zeigt alle Streams im Reorder-Camera-Szenario, in denen FPs aufgetreten sind. Das Szenario wurde mit dem MLDS-Verfahren fusioniert. Die FPs korrelieren zeitlich mit den tatsächlichen Anomalien.

aktiv manipuliert wurde. Eine Visualisierung der FPs im zeitlichen Verlauf eines Szenarios findet sich in Abbildung 9.7. Dort lässt sich erkennen, dass die FPs ausschließlich im Bereich zwischen 30 und 60 sowie 90 und 120 Sekunden auftreten, in denen auch Anomalien produziert wurden. Lässt man diese korrelierten Anomalien außen vor, so steigt die Präzision des Szenarios von ca. 20% auf 100%.

Dieser Umstand führt im RC, in dem keine der normalen Anomalien erkannt wurde, dazu, dass trotzdem in den richtigen Zeitabschnitten Anomalien erkannt wurden, auch wenn diese eigentlich als FPs angesehen wurden. Dies lässt sich der Abbildung 9.8 entnehmen.

9.7 Fazit der Fusion

Insgesamt scheint die Datenfusion wenig Potential zu besitzen, da kein Fusionsverfahren an die Leistung des besten durchschnittlichen F1-Scores aus der Anomalieerkennung her-

ankam. Dazu muss jedoch erwähnt werden, dass bspw. die MLDS-Fusion Probleme mit der Identifizierung des Anomalie-Datenstroms im RC-Szenario hatte. Da der F1-Score dort bei 0 lag, zieht dies den Schnitt bei vier Anomalieszenarien erheblich nach unten. Dieser Umstand führt dazu, dass der durchschnittliche F1-Score der MLDS-Fusion ohne das RC-Szenario mit ca. 64% sogar über der reinen Anomalieerkennung liegen würde.

Insgesamt muss positiv hervorgehoben werden, dass die Fusion abseits der Metriken wertvolle Ergebnisse liefern kann, so dass mit einer Intervallgröße von 0,05 Sekunden und der MLDS-Fusion jeder Zeitpunkt mit Anomalien identifiziert werden konnte, während im Baseline-Szenario (ohne Anomalien) keine falschen Klassifizierungen aufgetreten sind. Damit konnte eine Konfiguration ermittelt werden, die in allen Szenarien gute Ergebnisse liefern konnte. Negativ muss jedoch betrachtet werden, dass der Anomalie-Datenstrom in den meisten Szenarien nicht korrekt identifiziert werden konnte, was zielgerichtete Gegenmaßnahmen erschwert. Außerdem konnten die zeitlichen Grenzen der Anomalien nicht immer genau identifiziert werden, sodass Bereiche kurz vor oder nach der eigentlichen Anomalie fälschlicherweise mit zu der Anomalie gezählt wurden. Keiner dieser negativen Aspekte würde jedoch dazu führen, dass sich ein Fahrer bei der Fahrt mit Gegenmaßnahmen konfrontiert sehen würde, wenn diese nicht notwendig bzw. berechtigt gewesen wären.

10 Fazit und Ausblick

In diesem Kapitel wird das Fazit für die Arbeit gezogen. Es wird darauf eingegangen werden, welche Herausforderungen bewältigt wurden und welche Erkenntnisse gewonnen werden konnten. Außerdem wird ein Ausblick gegeben, wie die Erkenntnisse genutzt werden können, um ein Produktivsystem für den realen Einsatz der Anomalieerkennung zu entwickeln. Dazu gehört auch, welche Punkte verbessert oder in zukünftigen Projekten umgesetzt werden könnten. Zuletzt sollen die Forschungsfragen beantwortet werden, die zu Beginn dieser Arbeit gestellt wurden.

10.1 Anomalieerkennung

Das Framework, welches für die Erstellung von Trainings- und Testdaten in Form von Netzwerkaufzeichnungen eines Fahrzeugs sowie deren Verarbeitung und Analyse entwickelt wurde, bietet viele Möglichkeiten, anhand derer die Netzwerkaufzeichnung und die Anomalieerkennung konfiguriert werden können.

Während der Umsetzung des Frameworks wurden einige Schritte unternommen, um das Laufzeitverhalten der Software zu verbessern. So konnte an vielen Stellen parallelisiert werden, um die Datenverarbeitung zu beschleunigen, während an anderer Stelle das mehrfache Berechnen von Netzwerkmetriken verhindert wurde.

Mit dem implementierten Framework wurde anhand von vier Testszenarien untersucht, welchen Einfluss verschiedene Netzwerkmetriken, Intervallgrößen und Algorithmuskonfigurationen auf die Erkennungsleistung haben. Es konnte ermittelt werden, dass die Algorithmen HBO, MS und KM am besten für den Einsatz in der Anomalieerkennung geeignet sind. Alle untersuchten Konfigurationen der Algorithmen sind für diesen Einsatzzweck geeignet. Die Algorithmen AE und SVM wurden in diesem Zuge als ungeeignet für die Anomalieerkennung befunden. Da alle Submengen der Netzwerkmetriken für die Datenfusion eingesetzt werden, wurde keine explizite Analyse zu diesen durchgeführt.

Zuletzt kann die Intervallgröße auf einen Wert von 0,05 Sekunden festgelegt werden, da dieser durchschnittlich die besten Ergebnisse geliefert hat.

Über alle Szenarien und Algorithmen hinweg konnte festgestellt werden, dass das Aufkommen von FPs ein größeres Problem darstellt als die Erkennung von Anomalien. Diese Erkenntnis spielte für die Implementation und Konfiguration der Fusionsverfahren eine wichtige Rolle.

10.2 Datenfusion

Die Datenfusion, mit dem Ziel, Erkennungsleistung und Präzision der Anomalieerkennung zu steigern, setzt auf insgesamt fünf Verfahren auf. Diese Verfahren fusionieren zunächst die verfügbaren Konfigurationen der Algorithmen, inklusive verschiedener Netzwerkmetriken, um anschließend die Switches und optional die Algorithmen zu fusionieren. Grundlegend gibt es dabei drei Arten von Fusion: Mehrheitsfusion, D-S-Fusion und Fusion über die Zeit. Daraus wurden fünf Verfahren abgeleitet, die relativ ähnliche Ergebnisse erzielen konnten.

Nach der Datenfusion mit dem MLDS-Verfahren sind die Evaluationswerte durchschnittlich besser als nach der Anomalieerkennung, weit vom Optimum entfernt. Jedoch weisen die Daten Merkmale auf, die auch ohne perfekte Klassifikation einen Wert mit sich bringen. So konnte erreicht werden, dass jeder Zeitpunkt, an dem Anomalien erzeugt wurden, identifiziert werden konnte und keine vermeintlichen Anomalie-Cluster erkannt wurden, die rein aus FPs bestanden.

Auf der anderen Seite ist es nicht ohne Fehler möglich, den Datenstrom zu identifizieren, in dem Anomalien auftreten. Dies erschwert gezielte Gegenmaßnahmen auf Basis von Datenströmen.

Wichtig zu beachten ist die Fenstergröße bei der Fusion über die Zeit. Diese limitiert die Erkennungsleistung bezogen auf die Länge von Anomalien. Bei der Wahl der Fenstergröße muss demnach berücksichtigt werden, wie lang die Anomalien sind, die erkannt werden sollen, da eine zu kleine Fenstergröße dafür sorgen kann, dass die Anomalien als Ausreißer betrachtet und entfernt werden.

10.3 Einsatz der Anomalieerkennung

Möchte man die Erkenntnisse dieser Arbeit nutzen, um die Anomalieerkennung und Fusionsverfahren in einem realen Fahrzeug zu etablieren, so kann die grundlegende Funktionsweise übernommen werden.

Zunächst muss betrachtet werden, woher die Trainingsdaten für die Anomalieerkennung stammen. Dies wäre einfach möglich, da schlicht der normale Netzwerkverkehr in einer breiten Anzahl normaler Fahrsituationen aufgezeichnet werden kann. Die Aufzeichnung könnte dabei, wie in dieser Arbeit, an zentralen Knotenpunkten im Netzwerk, also den Switches, erfolgen. Egal, ob die Auswertung der Daten dezentral auf den Switches passiert und hinterher an einem zentralen Punkt fusioniert wird, oder die Rohdaten direkt zentral ausgewertet werden: Es muss immer beachtet werden, dass auch der Datenstrom der Anomalieerkennung von Angriffen betroffen sein könnte. Dieses Problem wurde aufgrund der Komplexität in dieser Arbeit nicht betrachtet.

Für die Konfiguration der verschiedenen Parameter sollten Testdaten erstellt werden, indem verschiedene Angriffsszenarien entweder in der realen Umgebung oder in einem OMNeT++-Nachbau durchgeführt werden, während der Netzwerkverkehr aufgezeichnet wird. Dabei ist es besonders wichtig, die Pakete zu markieren, die im Rahmen des Testlaufs manipuliert wurden. Für die Ermittlung der Parameter kann das im Rahmen dieser Arbeit erstellte Framework genutzt werden, da mit den Konfigurationsdateien beliebige Netzwerke untersucht werden könnten.

Die Fusionsverfahren können übernommen werden, doch es muss beachtet werden, dass die Fusion über die Zeit für jede Dateninstanz sowohl die Vergangenheit als auch die Zukunft miteinbezieht. Dies würde bei der derzeitigen Funktionsweise unweigerlich zu einer Verzögerung in der Auswertung führen, sodass bei einer Fenstergröße von einer Sekunde mindestens eine Sekunde Verzögerung von der Datensammlung bis zum Ergebnis erzeugt werden würde.

10.4 Beantwortung der Forschungsfragen

In dieser abschließenden Sektion soll ein Blick zurück auf die in der Einleitung (Kapitel 1) vorgestellten und in der Zielsetzung (Kapitel 3) konkretisierten Forschungsfragen geworfen werden. Die erste und zugleich am leichtesten zu beantwortende Frage zielt auf

die Eignung der sieben in der Arbeit genutzten Algorithmen für die Anomalieerkennung in Fahrzeugnetzwerken ab. Der EE-Algorithmus ist nicht geeignet, da sich teilweise zu wenig Varianz in den Datensätzen befindet und die zugrunde liegende Kovarianzmatrix teilweise singulär wird, was zu numerischen Instabilitäten führen würde und daher von der genutzten Bibliothek [11] nicht erlaubt ist. Die beiden Algorithmen AE und SVM sind aufgrund ihrer hohen FPR eher ungeeignet für den Einsatz in der Anomalieerkennung. Bedingt geeignet ist der IF-Algorithmus, da dieser zwar die wenigsten FPs liefert, aber dafür nur in einem der vier Szenarien Anomalien erkennen konnte. Die verbleibenden vier Algorithmen KM, MS und HBO eignen sich gut für die Anomalieerkennung, da sie nahezu alle Anomalien erkennen konnten und eine vertretbare Menge FPs erzeugen.

Der zweite Punkt, den es im Verlauf der Arbeit zu beachten galt, bezog sich auf Maßnahmen für eine möglichst effiziente Durchführung der Anomalieerkennung. Hier sind auf der einen Seite die zahlreichen Parallelisierungen zu erwähnen, mit denen die Berechnung beschleunigt wurde, und andererseits die Steigerung der Effizienz durch das Weglassen nicht notwendiger Berechnungen bei der Berechnung der Netzwerkmetriken. Nicht zu vernachlässigen ist hier auch die Datenstromseparierung, bei welcher durch die Einführung der Divide- und Merge-Phase ein Großteil der Rechenzeit eingespart werden konnte.

Das zentrale Problem der Fehlalarme konnte durch eine Analyse der FP-Datenströme vor und nach der Fusion angegangen werden. Die Daten zeigen nach der Fusion eine starke Korrelation mit den tatsächlich auftretenden Anomalien und sind damit weniger kritisch als unkorrelierte FPs, wie sie vor der Fusion aufgetreten sind.

Der letzte Punkt, welcher zugleich eine Zusammenfassung der Arbeit darstellt, bezog sich auf das mögliche Potenzial von Datenfusion. Negativ fällt hier ins Gewicht, dass die Ergebnisse vor der Fusion (gemessen am F1-Score) besser waren als die Ergebnisse nach der Fusion. Positiv fällt die Verteilung der FPs ins Gewicht, welche in den betrachteten Szenarien keine Gegenmaßnahmen ohne zugrundeliegende Anomalien ausgelöst hätten.

Insgesamt zeigt diese Arbeit, dass durch gezielte Auswahl von Algorithmen, durchdachte Systemkonfigurationen und den Einsatz von Datenfusion eine robuste Anomalieerkennung im Fahrzeugumfeld möglich ist, auch wenn noch Optimierungspotenziale als Grundlage für weiterführende Arbeiten bestehen.

Literaturverzeichnis

[1] Bannour, Fetia ; Souihi, Sami ; Mellouk, Abdelhamid: Distributed SDN Control: Survey, Taxonomy, and Challenges. In: *IEEE Communications Surveys Tutorials* 20 (2018), Nr. 1, S. 333–354

[2] Bhuyan, Monowar H. ; Bhattacharyya, D. K. ; Kalita, J. K.: Network Anomaly Detection: Methods, Systems and Tools. In: *IEEE Communications Surveys Tutorials* 16 (2014), Nr. 1, S. 303–336

[3] Bishop, Christopher M.: *Pattern Recognition and Machine Learning*. New York : Springer, 2006. – ISBN 9780387310732

[4] Carreira-Perpinan, Miguel A.: Gaussian Mean-Shift Is an EM Algorithm. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29 (2007), Nr. 5, S. 767–776

[5] Chandola, Varun ; Banerjee, Arindam ; Kumar, Vipin: Anomaly Detection: A Survey. In: *ACM Computing Surveys* 41, Nr. 3, S. 1–10. – URL https://doi.org/10.1145/1541880.1541882. – ISSN 0360-0300

[6] Checkoway, Stephen ; McCoy, Damon ; Kantor, Brian ; Anderson, Danny ; Shacham, Hovav ; Savage, Stefan ; Koscher, Karl ; Czeskis, Alexei ; Roesner, Franziska ; Kohno, Tadayoshi: Comprehensive experimental analyses of automotive attack surfaces. In: *20th USENIX security symposium (USENIX Security 11)*, 2011

[7] Chen, Zhaomin ; Yeo, Chai K. ; Lee, Bu S. ; Lau, Chiew T.: Autoencoder-based network anomaly detection. In: *2018 Wireless Telecommunications Symposium (WTS)*, 2018, S. 1–5

[8] Development-Team, INET-Framework: *INET Framework - What Is INET Framework?* https://inet.omnetpp.org/Introduction.html. 2025. – Accessed: 2025-02-10

[9] Development-Team, OMNeT++: *What is OMNeT++?* https://omnetpp.org/intro/. 2025. – Accessed: 2025-02-10

[10] Development-Team, TensorFlow: *Why TensorFlow?* https://www.tensorflow.org/about. 2025. – Accessed: 2025-02-20

[11] Development-Team scikit-learn: *About us?* https://scikit-learn.org/stable/about.html. 2025. – Accessed: 2025-02-20

[12] Fessi, B.A. ; Abdallah, S. B. ; Djemaiel, Y. ; Boudriga, N.: A Clustering Data Fusion Method for Intrusion Detection System. In: *2011 IEEE 11th International Conference on Computer and Information Technology*, 2011, S. 539–545

[13] Goldstein, Markus ; Dengel, Andreas: Histogram-based outlier score (hbos): A fast unsupervised anomaly detection algorithm. In: *KI-2012: poster and demo track* 1 (2012), S. 59–63

[14] Hall, D.L. ; Llinas, J.: An introduction to multisensor data fusion. In: *Proceedings of the IEEE* 85 (1997), Nr. 1, S. 6–23

[15] He, Ke ; Kim, Dan D. ; Asghar, Muhammad R.: Adversarial Machine Learning for Network Intrusion Detection Systems: A Comprehensive Survey. In: *IEEE Communications Surveys Tutorials* 25 (2023), Nr. 1, S. 538–566

[16] IEEE: IEEE 802.1CB: Frame Replication and Elimination for Reliability. In: *IEEE Standard*, URL https://ieeexplore.ieee.org/document/8091139, 2017

[17] Jiang, Junhui ; Li, Yuting ; Hong, Seung H. ; Xu, Aidong ; Wang, Kai: A Time-sensitive Networking (TSN) Simulation Model Based on OMNET++. In: *2018 IEEE International Conference on Mechatronics and Automation (ICMA)*, 2018, S. 643–648

[18] Li, Shi ; Wang, Huaqing ; Song, Liuyang ; Wang, Pengxin ; Cui, Lingli ; Lin, Tianjiao: An adaptive data fusion strategy for fault diagnosis based on the convolutional neural network. In: *Measurement* 165 (2020), S. 108122. – URL https://www.sciencedirect.com/science/article/pii/S0263224120306606. – ISSN 0263-2241

[19] Meyer, Philipp ; Häckel, Timo ; Lübeck, Teresa ; Korf, Franz ; Schmidt, Thomas C.: A Framework for the Systematic Assessment of Anomaly Detectors in Time-Sensitive Automotive Networks. In: *2024 IEEE Vehicular Networking Conference (VNC)*, 2024, S. 57–64

[20] Meyer, Philipp and Häckel, Timo and Lübeck, Teresa and Korf, Franz and Schmidt, Thomas C.: *NIDSDatasetCreation Car-Simulation Sources*. https://github.com/CoRE-RG/NIDSDatasetCreation/tree/main/simulations/car. 2025. – Accessed: 2025-04-19

[21] Mitchell, H.B.: *Data Fusion: Concepts and Ideas*. 2nd edition. Heidelberg, New York, Dordrecht, London : Springer-Verlag Berlin Heidelberg, 2012. – ISBN 9783642272219

[22] Murphy, Kevin P.: *Machine Learning: A Probabilistic Perspective*. Massachusetts Institute of Technology, 2012. – ISBN 9780262018029

[23] Schuhmacher, Wilhelm: *Methoden zur Anomalieerkennung in TSN basierten Fahrzeugnetzwerken*, Hochschule für angewandte Wissenschaften Hamburg, Mastersthesis, 2023

[24] Sentz, Kari ; Ferson, Scott: Combination of Evidence in Dempster-Shafer Theory / Sandia National Laboratories. Albuquerque, New Mexico, 2002 (SAND2002-0835). – Technical Report

[25] Soltani, Mahdi ; Siavoshani, Mahdi J. ; Jahangir, Amir H.: A content-based deep intrusion detection system. In: *International Journal of Information Security* 21 (2022), Nr. 3, S. 547–562. – URL https://doi.org/10.1007/s10207-021-00567-2. – ISSN 1615-5270

[26] Sommer, Florian ; Dürrwang, Jürgen ; Kriesten, Reiner: Survey and Classification of Automotive Security Attacks. In: *Information* 10 (2019), Nr. 4. – URL https://www.mdpi.com/2078-2489/10/4/148. – ISSN 2078-2489

[27] Talpur, Anum ; Gurusamy, Mohan: Machine Learning for Security in Vehicular Networks: A Comprehensive Survey. In: *IEEE Communications Surveys Tutorials* 24 (2022), Nr. 1, S. 346–379

[28] Vishwakarma, Monika ; Kesswani, Nishtha: A new two-phase intrusion detection system with Naïve Bayes machine learning for data classification and elliptic envelop method for anomaly detection. In: *Decision Analytics Journal* 7 (2023), S. 100233. – URL https://www.sciencedirect.com/science/article/pii/S2772662223000735. – ISSN 2772-6622

[29] Wireshark Foundation: *Editcap – Manual Page*. https://www.wireshark.org/docs/man-pages/editcap.html. 2025. – Accessed: 2025-04-08

[30] Wireshark Foundation: *Mergecap – Manual Page*. https://www.wireshark.org/docs/man-pages/mergecap.html. 2025. – Accessed: 2025-04-08

[31] Wireshark Foundation: *TShark – Manual Page*. https://www.wireshark.org/docs/man-pages/tshark.html. 2025. – Accessed: 2025-04-08

[32] Xiao, Yanghui ; Shi, Zelin: Application of multi-sensor data fusion technology in target recognition. In: *2011 3rd International Conference on Advanced Computer Control*, 2011, S. 441–444

[33] Xu, Dong ; Wang, Yanjun ; Meng, Yulong ; Zhang, Ziying: An Improved Data Anomaly Detection Method Based on Isolation Forest. In: *2017 10th International Symposium on Computational Intelligence and Design (ISCID)* Bd. 2, 2017, S. 287–291

[34] Zhang, Lan ; Leung, Henry ; Chan, Keith C.: Information fusion based smart home control system and its application. In: *IEEE Transactions on Consumer Electronics* 54 (2008), Nr. 3, S. 1157–1165

A Anhang

In den folgenden Anhängen sollen die Ergebnisse der Anomalieerkennung tabellarisch dargestellt werden. Zunächst folgen die Tabellen mit den Ergebnissen der Anomalieerkennung, im Anschluss daran die Tabellen mit den Ergebnissen der Datenfusion. Um Platz zu sparen, wurden einige Abkürzungen verwendet, die hier aufgelistet sind:

- P(A): Wahrscheinlichkeit für das Auftreten einer Anomalie.
- Intvl.: Intervallgröße für die Berechnung der Netzwerkmetriken.
- Algo.: Der für die Anomalieerkennung verwendete Algorithmus.
- Konfig.: Die Konfiguration des Algorithmus oder der Datenfusion.
- Prec.: Die Präzision der Anomalieerkennung oder der Datenfusion.
- Cont.: Der Algorithmus-Parameter Contamination.
- BEF: Der Algorithmus-Parameter Border Enlargement Factor.
- B: Die Netzwerkmetrik der Bandbreite.
- FS: Die Netzwerkmetrik der durchschnittlichen Frame-Größe.
- FS: Die Netzwerkmetrik des durchschnittlichen Frame-Abstands.
- J: Die Netzwerkmetrik des Jitters.
- WS: Die Konfiguration der Fenstergröße (Window Size).
- A: Die Konfiguration der Anomalie-Masse.
- N: Die Konfiguration der Normal-Masse.
- a: Die Konfiguration des a-Values für die Konstruktion der Massen.

A.1 Verwendete Hilfsmittel

In der Tabelle A.1 sind die im Rahmen der Bearbeitung des Themas der Bachelorarbeit verwendeten Werkzeuge und Hilfsmittel aufgelistet.

Tabelle A.1: Verwendete Hilfsmittel und Werkzeuge

Tool	**Verwendung**
LaTeX	Textsatz- und Layout-Werkzeug, verwendet zur Erstellung dieses Dokuments.
OMNeT++ und INET	Frameworks für die präzise Simulation des Echtzeit-Ethernet- und Datenverkehrs eines Fahrzeugnetzwerks. [8][9]
NIDSDataset-Creation- und PyNADS-Framework	Frameworks zur Simulation und Bereitstellung Anomalieerkennung. [19][20]
TShark, Mergecap und Editcap	Frameworks für die Analyse und Bearbeitung von Netzwerkaufzeichnungen. [31][30][29]

A.2 Tabellen - Anomalieerkennung

Tabelle A.2: Ergebnisse der Anomalieerkennung für das Eliminate Auto Brake Szenario. Für jede Kombination wird jeweils nur die beste Algorithmus-Konfiguration angezeigt.

Kombination					Ergebnis				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Metriken**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.1	0.05	AE	None: 0.0	FS-FG-J	0.03835	**0.99947**	0.07386	0.13372	140565
0.1	0.1	AE	None: 0.0	B-FS-FG-J	0.02868	**1**	0.05577	0.14321	191394
0.1	1.0	AE	None: 0.0	B-FS-FG	0.01981	**1**	0.03884	0.14809	285060
0.25	0.05	AE	None: 0.0	FS-FG-J	0.03859	**1**	0.07432	0.13367	140499
0.25	0.1	AE	None: 0.0	FS-FG-J	0.02862	**1**	0.05564	0.14355	191856
0.25	1.0	AE	None: 0.0	B-FG-J	0.02003	**0.97917**	0.03925	0.14338	276000
0.5	0.05	AE	None: 0.0	FG-J	0.03862	**1**	0.07438	0.13341	140233.5
0.5	0.1	AE	None: 0.0	B-FS-FG-J	0.02838	**0.99947**	0.05519	0.14449	193113
0.5	1.0	AE	None: 0.0	B-FG	0.02039	**1**	0.03996	0.14379	276780
0.1	0.05	HBO	Cont.: 0.0001	FS-FG	**0.96349**	**0.99519**	**0.97908**	**0.0002**	211.5
0.1	0.1	HBO	Cont.: 0.0001	FS-FG	**0.89629**	**1**	**0.94531**	**0.00049**	654
0.1	1.0	HBO	Cont.: 0.0001	B-FS	0.28487	**1**	0.44342	0.00751	14460
0.25	0.05	HBO	Cont.: 0.0001	FS-FG	**0.95796**	**1**	**0.97853**	**0.00024**	247.5
0.25	0.1	HBO	Cont.: 0.0001	FS-FG	**0.89544**	**1**	**0.94483**	**0.00049**	660
0.25	1.0	HBO	Cont.: 0.0001	B-FS	0.28277	**1**	0.44087	0.00759	14610
0.5	0.05	HBO	Cont.: 0.0001	FS-FG	**0.94872**	**1**	**0.97369**	**0.00029**	304.5
0.5	0.1	HBO	Cont.: 0.0001	FS-FG	**0.89439**	**0.99947**	**0.94401**	**0.0005**	666
0.5	1.0	HBO	Cont.: 0.0001	FS-FG	0.2836	**1**	0.44189	0.00756	14550
0.1	0.05	IF	Cont.: 0.0001	B-FS	0	0	0	**0.0001**	109.5
0.1	0.1	IF	Cont.: 0.0001	FS-J	0.00763	0.00053	0.00099	**0.00029**	390
0.1	1.0	IF	Cont.: 0.0001	B-FS	0	0	0	**0.00019**	360
0.25	0.05	IF	Cont.: 0.0001	B-FS	0	0	0	**0.00011**	111
0.25	0.1	IF	Cont.: 0.0001	FS-J	0.00758	0.00053	0.00099	**0.00029**	393
0.25	1.0	IF	Cont.: 0.0001	B-FS	0	0	0	**0.0002**	390
0.5	0.05	IF	Cont.: 0.0001	B-FS	0	0	0	**0.00011**	111
0.5	0.1	IF	Cont.: 0.0001	FS-J	0.01504	0.00106	0.00199	**0.00029**	393
0.5	1.0	IF	Cont.: 0.0001	B-FS	0	0	0	**0.00017**	330
0.1	0.05	KM1	BEF: 0.8	FS-J	0.24962	0.70928	0.36928	0.01138	11958
0.1	0.1	KM1	BEF: 0.8	FS-J	0.12426	0.66826	0.20955	0.01992	26619
0.1	1.0	KM1	BEF: 0.8	B-FG	0.02794	**1**	0.05435	0.10412	200430
0.25	0.05	KM1	BEF: 0.8	FS-J	0.2639	0.7625	0.3921	0.01141	11995.5
0.25	0.1	KM1	BEF: 0.8	FS-J	0.12405	0.66826	0.20926	0.01996	26670
0.25	1.0	KM1	BEF: 0.8	B-FG	0.02791	**1**	0.05431	0.10422	200610
0.5	0.05	KM1	BEF: 0.8	FS-J	0.28207	0.83946	0.42226	0.01145	12037.5
0.5	0.1	KM1	BEF: 0.8	FS-J	0.1237	0.66773	0.20872	0.01997	26694
0.5	1.0	KM1	BEF: 0.8	B-FG	0.02792	**1**	0.05433	0.10417	200520
0.1	0.05	MS	BEF: 2.0	FS-FG	0.75485	**0.99893**	**0.85991**	**0.00173**	1819.5
0.1	0.1	MS	BEF: 2.0	FG-J	**0.90057**	**1**	**0.94769**	**0.00047**	624
0.1	1.0	MS	BEF: 2.0	B-FS-FG	0.18972	**1**	0.31894	0.01278	24600
0.25	0.05	MS	BEF: 2.0	FG-J	**0.9393**	**1**	**0.9687**	**0.00035**	364.5
0.25	0.1	MS	BEF: 2.0	FG-J	**0.89331**	**1**	**0.94365**	**0.00051**	675
0.25	1.0	MS	BEF: 2.0	B-FS-FG	0.18842	**1**	0.31709	0.01289	24810
0.5	0.05	MS	BEF: 2.0	FG-J	**0.92832**	**1**	**0.96283**	**0.00041**	435
0.5	0.1	MS	BEF: 2.0	FG-J	**0.88139**	**0.99947**	**0.93672**	**0.00057**	759
0.5	1.0	MS	BEF: 2.0	B-FS-FG	0.18861	**1**	0.31736	0.01287	24780
0.1	0.05	SVM	Cont.: 0.0001	B-FS-FG-J	0.00825	**1**	0.01636	0.6415	674322
0.1	0.1	SVM	Cont.: 0.0001	B-FS-J	0.0047	**1**	0.00936	0.89521	1196445
0.1	1.0	SVM	Cont.: 0.0001	B-J	0.00298	**1**	0.00595	1	1924920
0.25	0.05	SVM	Cont.: 0.0001	B-FS-FG-J	0.0083	**1**	0.01645	0.64149	674287.5
0.25	0.1	SVM	Cont.: 0.0001	B-FS-J	0.0047	**1**	0.00936	0.89521	1196448
0.25	1.0	SVM	Cont.: 0.0001	B-J	0.00298	**1**	0.00595	1	1924920
0.5	0.05	SVM	Cont.: 0.0001	B-FS-FG-J	0.00829	**1**	0.01644	0.64149	674287.5
0.5	0.1	SVM	Cont.: 0.0001	B-FS-J	0.00469	**1**	0.00934	0.89521	1196451
0.5	1.0	SVM	Cont.: 0.0001	B-J	0.00298	**1**	0.00595	1	1924920

Tabelle A.3: Ergebnisse der Anomalieerkennung für das Inject LIDAR Szenario. Für jede Kombination wird jeweils nur die beste Algorithmus-Konfiguration angezeigt.

Kombination					Ergebnis				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Metriken**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.25	0.05	AE	None: 0.0	B-FG-J	0.02413	**1**	0.04711	0.14521	152901
0.25	0.1	AE	None: 0.0	B-FS-FG	0.01828	**1**	0.0359	0.152	203343
0.25	1.0	AE	None: 0.0	FS-FG-J	0.01336	**0.99219**	0.02636	0.14609	281430
0.5	0.05	AE	None: 0.0	FS-FG-J	0.02372	**1**	0.04634	0.14776	155580
0.5	0.1	AE	None: 0.0	B-FS-FG	0.018	**1**	0.03536	0.15436	206556
0.5	1.0	AE	None: 0.0	B-FS-FG-J	0.01354	**1**	0.02673	0.14512	279690
1.0	0.05	AE	None: 0.0	FS-FG-J	0.0236	**1**	0.04612	0.14851	156367.5
1.0	0.1	AE	None: 0.0	B-FG	0.01767	**1**	0.03473	0.15731	210468
1.0	1.0	AE	None: 0.0	FS-FG-J	0.01358	**0.98438**	0.02679	0.14251	274530
0.25	0.05	HBO	Cont.: 0.0001	FS-FG	0.2796	**1**	0.43701	0.00925	9739.5
0.25	0.1	HBO	Cont.: 0.0001	FS-FG	0.22726	**1**	0.37036	0.00962	12873
0.25	1.0	HBO	Cont.: 0.0001	FS-J	0.128	0.5	0.20382	0.00679	13080
0.5	0.05	HBO	Cont.: 0.0001	FS-FG	0.23092	**1**	0.3752	0.01196	12589.5
0.5	0.1	HBO	Cont.: 0.0001	FS-FG	0.2056	**1**	0.34108	0.01093	14628
0.5	1.0	HBO	Cont.: 0.0001	FS-FG	0.09162	**1**	0.16787	0.01975	38070
1.0	0.05	HBO	Cont.: 0.0001	FS-FG	0.21376	**1**	0.35223	0.01321	13903.5
1.0	0.1	HBO	Cont.: 0.0001	FS-FG	0.19824	**1**	0.33089	0.01144	15312
1.0	1.0	HBO	Cont.: 0.0001	B-FS	0.09639	**1**	0.17582	0.01869	36000
0.25	0.05	IF	Cont.: 0.0001	FS-FG-J	**0.85849**	**0.97262**	**0.912**	**0.00058**	606
0.25	0.1	IF	Cont.: 0.0001	B-FS	**0.94039**	0.5	0.65287	**0.00009**	120
0.25	1.0	IF	Cont.: 0.0001	FS-J	0.02308	0.04688	0.03093	**0.00396**	7620
0.5	0.05	IF	Cont.: 0.0001	B-FS-J	**0.884**	0.87698	**0.88048**	**0.00041**	435
0.5	0.1	IF	Cont.: 0.0001	B-FS	**0.95606**	0.5	0.65661	**0.00007**	87
0.5	1.0	IF	Cont.: 0.0001	FS-J	0.0232	0.07031	0.03488	0.0059	11370
1.0	0.05	IF	Cont.: 0.0001	FS-FG-J	**0.81681**	**0.96429**	**0.88444**	**0.00078**	817.5
1.0	0.1	IF	Cont.: 0.0001	B-FS	**0.94179**	0.5	0.65321	**0.00009**	117
1.0	1.0	IF	Cont.: 0.0001	FS-J	0.03526	0.08594	0.05	**0.00469**	9030
0.25	0.05	KM1	BEF: 0.8	FS-J	0.13205	**1**	0.2333	0.0236	24844.5
0.25	0.1	KM1	BEF: 0.8	FS-J	0.08501	**1**	0.1567	0.03046	40749
0.25	1.0	KM1	BEF: 0.8	B-FS-FG	0.01853	**1**	0.03639	0.10557	203370
0.5	0.05	KM1	BEF: 0.8	FS-J	0.1298	**1**	0.22978	0.02407	25341
0.5	0.1	KM1	BEF: 0.8	FS-J	0.08232	**1**	0.15212	0.03154	42204
0.5	1.0	KM1	BEF: 0.8	B-FS-FG	0.01784	**1**	0.03506	0.10966	211350
1.0	0.05	KM1	BEF: 0.8	FS-J	0.13723	**1**	0.24134	0.02257	23764.5
1.0	0.1	KM1	BEF: 0.8	FS-J	0.08575	**1**	0.15795	0.03017	40368
1.0	1.0	KM1	BEF: 0.8	B-FS-FG	0.01747	**1**	0.03433	0.11213	216000
0.25	0.05	MS	BEF: 2.0	FS-FG	0.23651	**1**	0.38254	0.01159	12202.5
0.25	0.1	MS	BEF: 2.0	FS-J	0.19211	**1**	0.32231	0.0119	15921
0.25	1.0	MS	BEF: 2.0	B-FS-FG	0.0806	**1**	0.14918	0.02274	43800
0.5	0.05	MS	BEF: 2.0	FS-J	0.2008	**1**	0.33444	0.01429	15045
0.5	0.1	MS	BEF: 2.0	FS-J	0.18921	**1**	0.3182	0.01212	16224
0.5	1.0	MS	BEF: 2.0	B-FS-FG	0.07887	**1**	0.1462	0.02327	44850
1.0	0.05	MS	BEF: 2.0	FS-J	0.22392	**1**	0.36591	0.01244	13101
1.0	0.1	MS	BEF: 2.0	FS-J	0.21153	**1**	0.3492	0.01055	14112
1.0	1.0	MS	BEF: 2.0	B-FS-FG	0.08076	**1**	0.14945	0.02269	43710
0.25	0.05	SVM	Cont.: 0.0001	B-FS-FG-J	0.00554	**1**	0.01102	0.64457	678684
0.25	0.1	SVM	Cont.: 0.0001	B-FS-J	0.00314	**1**	0.00627	0.89721	1200243
0.25	1.0	SVM	Cont.: 0.0001	B-J	0.00199	**1**	0.00397	1	1926360
0.5	0.05	SVM	Cont.: 0.0001	B-FS-FG-J	0.00553	**1**	0.011	0.64569	679854
0.5	0.1	SVM	Cont.: 0.0001	B-FS-J	0.00314	**1**	0.00626	0.89769	1201212
0.5	1.0	SVM	Cont.: 0.0001	B-J	0.00199	**1**	0.00397	1	1927260
1.0	0.05	SVM	Cont.: 0.0001	B-FS-FG-J	0.00552	**1**	0.01098	0.64689	681099
1.0	0.1	SVM	Cont.: 0.0001	B-FS-J	0.00314	**1**	0.00626	0.89803	1201509
1.0	1.0	SVM	Cont.: 0.0001	B-J	0.00199	**1**	0.00397	1	1926360

Tabelle A.4: Ergebnisse der Anomalieerkennung für das Delay Manual Steer Szenario. Für jede Kombination wird jeweils nur die beste Algorithmus-Konfiguration angezeigt.

Kombination					Ergebnis				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Metriken**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.1	0.05	AE	None: 0.0	FG-J	0.03355	0.88447	0.06465	0.13626	143232
0.1	0.1	AE	None: 0.0	B-FG-J	0.02515	0.91936	0.04896	0.15079	201507
0.1	1.0	AE	None: 0.0	B-FG	0.00836	0.42105	0.01639	0.14803	284820
0.25	0.05	AE	None: 0.0	FG-J	0.03474	0.91115	0.06693	0.13581	142753.5
0.25	0.1	AE	None: 0.0	FS-FG-J	0.02551	0.90716	0.04962	0.14666	195987
0.25	1.0	AE	None: 0.0	FS-FG	0.00867	0.46316	0.01701	0.15695	301980
0.5	0.05	AE	None: 0.0	B-FG-J	0.03422	0.91753	0.06598	0.13891	146008.5
0.5	0.1	AE	None: 0.0	B-FS-FG-J	0.02542	0.93206	0.04949	0.15114	201972
0.5	1.0	AE	None: 0.0	B-FG	0.00813	0.38542	0.01593	0.14069	270690
0.1	0.05	HBO	Cont.: 0.0001	FS-FG-J	0.48358	0.88767	0.62608	0.00507	5329.5
0.1	0.1	HBO	Cont.: 0.0001	B-FS-FG-J	0.44272	0.87745	0.58851	**0.00467**	6246
0.1	1.0	HBO	Cont.: 0.0001	B-FS	0.18704	0.66842	0.29229	0.00861	16560
0.25	0.05	HBO	Cont.: 0.0001	FS-FG-J	0.4617	0.91088	0.6128	0.0057	5988
0.25	0.1	HBO	Cont.: 0.0001	B-FS-FG-J	0.43313	0.90716	0.58632	0.00502	6714
0.25	1.0	HBO	Cont.: 0.0001	B-FS	0.19651	0.71053	0.30787	0.00861	16560
0.5	0.05	HBO	Cont.: 0.0001	FS-FG	0.49671	0.74302	0.5954	**0.00404**	4245
0.5	0.1	HBO	Cont.: 0.0001	FS-FG	0.48209	0.7431	0.5848	**0.00338**	4512
0.5	1.0	HBO	Cont.: 0.0001	B-FS	0.20588	0.76562	0.3245	0.00884	17010
0.1	0.05	IF	Cont.: 0.0001	FS-J	0.27483	0.03175	0.05692	**0.00045**	471
0.1	0.1	IF	Cont.: 0.0001	B-FS	0	0	0	**0.00016**	216
0.1	1.0	IF	Cont.: 0.0001	B-FS	0	0	0	**0.00011**	210
0.25	0.05	IF	Cont.: 0.0001	FS-J	0.31102	0.03831	0.06821	**0.00046**	478.5
0.25	0.1	IF	Cont.: 0.0001	B-FS	0	0	0	**0.00015**	207
0.25	1.0	IF	Cont.: 0.0001	B-FS	0	0	0	**0.00012**	240
0.5	0.05	IF	Cont.: 0.0001	FS-J	0.21474	0.03565	0.06115	**0.0007**	735
0.5	0.1	IF	Cont.: 0.0001	B-FS	0	0	0	**0.00017**	228
0.5	1.0	IF	Cont.: 0.0001	B-FS	0	0	0	**0.00011**	210
0.1	0.05	KM1	BEF: 0.8	FS-J	0.20285	0.69477	0.31402	0.0146	15349.5
0.1	0.1	KM1	BEF: 0.8	FS-J	0.12055	0.73793	0.20724	0.02278	30444
0.1	1.0	KM1	BEF: 0.8	B-FS-FG	0.02082	0.76842	0.04055	0.10704	205950
0.25	0.05	KM1	BEF: 0.8	FS-J	0.20098	0.70577	0.31287	0.01505	15820.5
0.25	0.1	KM1	BEF: 0.8	FS-J	0.12345	0.76499	0.21259	0.02299	30717
0.25	1.0	KM1	BEF: 0.8	B-FG	0.01984	0.73158	0.03864	0.10705	205980
0.5	0.05	KM1	BEF: 0.8	FS-J	0.19648	0.70311	0.30713	0.01542	16213.5
0.5	0.1	KM1	BEF: 0.8	FS-J	0.12291	0.7707	0.212	0.02326	31086
0.5	1.0	KM1	BEF: 0.8	B-FS-FG	0.02182	0.79688	0.04248	0.10695	205770
0.1	0.05	MS	BEF: 2.0	FG-J	0.48891	0.88767	0.63053	**0.00496**	5217
0.1	0.1	MS	BEF: 2.0	FG-J	0.46047	0.87745	0.60398	**0.00435**	5814
0.1	1.0	MS	BEF: 2.0	FG-J	0.16264	0.92105	0.27646	0.01405	27030
0.25	0.05	MS	BEF: 2.0	FG-J	0.46663	0.91141	0.61724	0.00559	5874
0.25	0.1	MS	BEF: 2.0	FG-J	0.44929	0.90716	0.60095	**0.00471**	6288
0.25	1.0	MS	BEF: 2.0	FG-J	0.15765	0.88947	0.26783	0.01408	27090
0.5	0.05	MS	BEF: 2.0	FG-J	0.44153	0.917	0.59606	0.00622	6540
0.5	0.1	MS	BEF: 2.0	FG-J	0.42864	0.92781	0.58638	0.00523	6990
0.5	1.0	MS	BEF: 2.0	FG-J	0.16088	0.91667	0.27372	0.01431	27540
0.1	0.05	SVM	Cont.: 0.0001	B-FS-FG-J	0.00827	**1**	0.01641	0.64133	674139
0.1	0.1	SVM	Cont.: 0.0001	B-FS-J	0.0047	**1**	0.00937	0.8952	1196319
0.1	1.0	SVM	Cont.: 0.0001	B-J	0.00295	**1**	0.00589	1	1924080
0.25	0.05	SVM	Cont.: 0.0001	B-FS-FG-J	0.00829	**1**	0.01645	0.64134	674130
0.25	0.1	SVM	Cont.: 0.0001	B-FS-J	0.0047	**1**	0.00937	0.8952	1196319
0.25	1.0	SVM	Cont.: 0.0001	B-J	0.00295	**1**	0.00589	1	1924080
0.5	0.05	SVM	Cont.: 0.0001	B-FS-FG-J	0.00829	**1**	0.01645	0.64135	674140.5
0.5	0.1	SVM	Cont.: 0.0001	B-FS-J	0.0047	**1**	0.00936	0.89521	1196319
0.5	1.0	SVM	Cont.: 0.0001	B-J	0.00298	**1**	0.00595	1	1924020

Tabelle A.5: Ergebnisse der Anomalieerkennung für das Reorder Camera Szenario. Für jede Kombination wird jeweils nur die beste Algorithmus-Konfiguration angezeigt.

Kombination					Ergebnis				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Metriken**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.1	0.05	AE	None: 0.0	FS-FG-J	0.01172	0.9429	0.02315	0.14254	150376.5
0.1	0.1	AE	None: 0.0	FS-FG-J	0.00869	**0.95246**	0.01723	0.15351	205638
0.1	1.0	AE	None: 0.0	B-FS-FG	0.00118	0.1875	0.00235	0.15797	304620
0.25	0.05	AE	None: 0.0	FG-J	0.01166	**0.96429**	0.02304	0.14645	154506
0.25	0.1	AE	None: 0.0	FS-FG-J	0.00869	**0.96513**	0.01723	0.15549	208350
0.25	1.0	AE	None: 0.0	B-FS-J	0.00082	0.125	0.00162	0.15218	293610
0.5	0.05	AE	None: 0.0	FG-J	0.01154	**0.9746**	0.0228	0.14957	157813.5
0.5	0.1	AE	None: 0.0	FG-J	0.00851	**0.9588**	0.01687	0.15778	211473
0.5	1.0	AE	None: 0.0	FS-FG	0.00248	0.39062	0.00492	0.15663	302250
0.1	0.05	HBO	Cont.: 0.0001	FS-FG-J	0.00674	0.02776	0.01084	0.00734	7741.5
0.1	0.1	HBO	Cont.: 0.0001	FG-J	0.00104	0.00634	0.00178	0.00862	11550
0.1	1.0	HBO	Cont.: 0.0001	FS-J	0.11132	0.9375	0.199	0.00745	14370
0.25	0.05	HBO	Cont.: 0.0001	B-FS-J	0.00546	0.03254	0.00935	0.01062	11203.5
0.25	0.1	HBO	Cont.: 0.0001	FG-J	0.00086	0.00634	0.00152	0.01036	13884
0.25	1.0	HBO	Cont.: 0.0001	FS-J	0.09428	**0.95312**	0.17159	0.00911	17580
0.5	0.05	HBO	Cont.: 0.0001	FS-FG-J	0.00516	0.03651	0.00904	0.01262	13312.5
0.5	0.1	HBO	Cont.: 0.0001	FS-FG-J	0.0007	0.00634	0.00126	0.01277	17112
0.5	1.0	HBO	Cont.: 0.0001	FS-J	0.07268	**0.96875**	0.13522	0.0123	23730
0.1	0.05	IF	Cont.: 0.0001	B-FG-J	0.00962	0.00397	0.00561	**0.00073**	772.5
0.1	0.1	IF	Cont.: 0.0001	B-FG-J	0.0086	0.00475	0.00612	**0.00077**	1038
0.1	1.0	IF	Cont.: 0.0001	B-FS	0	0	0	**0.00012**	240
0.25	0.05	IF	Cont.: 0.0001	B-FS-FG-J	0.00799	0.00476	0.00597	**0.00106**	1117.5
0.25	0.1	IF	Cont.: 0.0001	B-FG-J	0.01289	0.01109	0.01193	**0.0012**	1608
0.25	1.0	IF	Cont.: 0.0001	B-FS	0	0	0	**0.00008**	150
0.5	0.05	IF	Cont.: 0.0001	B-FS-FG-J	0.00336	0.00476	0.00394	**0.00253**	2667
0.5	0.1	IF	Cont.: 0.0001	B-FS	0.00893	0.00158	0.00269	**0.00025**	333
0.5	1.0	IF	Cont.: 0.0001	B-FS	0	0	0	**0.00022**	420
0.1	0.05	KM1	BEF: 0.8	FS-J	0.00411	0.03569	0.00738	0.01549	16344
0.1	0.1	KM1	BEF: 0.8	FS-J	0.0302	0.74168	0.05804	0.03366	45084
0.1	1.0	KM1	BEF: 0.8	FS-J	0.00279	0.46875	0.00554	0.16695	321930
0.25	0.05	KM1	BEF: 0.8	FS-J	0.01294	0.12778	0.02351	0.01746	18417
0.25	0.1	KM1	BEF: 0.8	FS-J	0.03619	0.67353	0.06869	0.02534	33954
0.25	1.0	KM1	BEF: 0.8	FS-J	0.00453	0.78125	0.00901	0.17083	329610
0.5	0.05	KM1	BEF: 0.8	FS-J	0.02541	0.26587	0.04638	0.01827	19276.5
0.5	0.1	KM1	BEF: 0.8	FS-J	0.04213	0.8019	0.08005	0.02575	34515
0.5	1.0	KM1	BEF: 0.8	FS-J	0.00509	0.90625	0.01012	0.17628	340170
0.1	0.05	MS	BEF: 2.0	FS-J	0.28049	0.86439	0.42355	**0.00398**	4194
0.1	0.1	MS	BEF: 2.0	FS-J	0.22205	0.89699	0.35597	**0.00444**	5949
0.1	1.0	MS	BEF: 2.0	FG-J	0.049	**0.95312**	0.0932	0.01842	35520
0.25	0.05	MS	BEF: 2.0	FS-J	0.20783	0.91032	0.3384	0.00622	6558
0.25	0.1	MS	BEF: 2.0	FS-J	0.18526	0.93185	0.30907	0.00579	7758
0.25	1.0	MS	BEF: 2.0	FS-J	0.04583	**0.95312**	0.08746	0.01975	38100
0.5	0.05	MS	BEF: 2.0	FS-J	0.15982	0.92302	0.27246	0.00869	9171
0.5	0.1	MS	BEF: 2.0	FS-J	0.1436	0.91601	0.24828	0.00772	10341
0.5	1.0	MS	BEF: 2.0	FS-J	0.03837	**0.96875**	0.07381	0.02416	46620
0.1	0.05	SVM	Cont.: 0.0001	FS-FG-J	0.00264	**1**	0.00527	0.67628	713464.5
0.1	0.1	SVM	Cont.: 0.0001	FG-J	0.00155	**1**	0.00309	0.91245	1222299
0.1	1.0	SVM	Cont.: 0.0001	B-J	0.00099	**1**	0.00199	1	1928230
0.25	0.05	SVM	Cont.: 0.0001	FS-FG-J	0.00264	**1**	0.00526	0.67698	714228
0.25	0.1	SVM	Cont.: 0.0001	FG-J	0.00154	**1**	0.00308	0.91373	1224387
0.25	1.0	SVM	Cont.: 0.0001	B-J	0.00099	**1**	0.00199	1	1929420
0.5	0.05	SVM	Cont.: 0.0001	FS-FG-J	0.00263	**1**	0.00525	0.6785	715882.5
0.5	0.1	SVM	Cont.: 0.0001	FG-J	0.00154	**1**	0.00308	0.91482	1226130
0.5	1.0	SVM	Cont.: 0.0001	B-J	0.00099	**1**	0.00199	1	1929720

A.3 Tabellen - Datenfusion

A.3.1 Tabellen: Mehrheitsfusion

Tabelle A.6: Ergebnisse der Datenfusion für das Baseline Szenario und der Methode Threshold.

Kombination				**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.0	0.05	HBO		0	0	0	**0.00024**	78
0.0	0.1	HBO		0	0	0	**0.00046**	207
0.0	1.0	HBO		0	0	0	**0.0028**	1800
0.0	0.05	IF		0	0	0	**0**	1.5
0.0	0.1	IF		0	0	0	**0.00001**	3
0.0	1.0	IF		0	0	0	**0.00065**	420
0.0	0.05	KM1		0	0	0	0.0525	17203.5
0.0	0.1	KM1		0	0	0	0.0303	13689
0.0	1.0	KM1		0	0	0	0.08419	54120
0.0	0.05	MS		0	0	0	**0.0001**	33
0.0	0.1	MS		0	0	0	**0.00002**	9
0.0	1.0	MS		0	0	0	**0.00112**	720

Tabelle A.7: Ergebnisse der Datenfusion für das Eliminate Auto Brake Szenario und der Methode Threshold.

Kombination				**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.1	0.05	HBO		**0.90288**	**1**	**0.94896**	**0.00062**	202.5
0.1	0.1	HBO		0.7858	**1**	**0.88006**	**0.00114**	516
0.1	1.0	HBO		0.20063	**1**	0.3342	0.0118	7650
0.25	0.05	HBO		**0.8863**	**1**	**0.93972**	**0.00074**	241.5
0.25	0.1	HBO		0.78385	**1**	**0.87883**	**0.00115**	522
0.25	1.0	HBO		0.19512	**1**	0.32653	0.01222	7920
0.5	0.05	HBO		**0.86295**	**1**	**0.92643**	**0.00091**	298.5
0.5	0.1	HBO		0.78218	**1**	**0.87778**	**0.00116**	528
0.5	1.0	HBO		0.19814	**1**	0.33075	0.01199	7770
0.1	0.05	IF		0	0	0	**0.00001**	3
0.1	0.1	IF		0	0	0	**0.00001**	6
0.1	1.0	IF		0	0	0	**0.00014**	90
0.25	0.05	IF		0	0	0	**0.00001**	4.5
0.25	0.1	IF		0	0	0	**0.00002**	9
0.25	1.0	IF		0	0	0	**0.00019**	120
0.5	0.05	IF		0	0	0	**0.00001**	3
0.5	0.1	IF		0	0	0	**0.00002**	9
0.5	1.0	IF		0	0	0	**0.00014**	90
0.1	0.05	KM1		0.09839	**1**	0.17914	0.05253	17251.5
0.1	0.1	KM1		0.12334	**1**	0.21959	0.02963	13455
0.1	1.0	KM1		0.0363	**1**	0.07006	0.07863	50970
0.25	0.05	KM1		0.09828	**1**	0.17897	0.05259	17272.5
0.25	0.1	KM1		0.12269	**1**	0.21857	0.02981	13536
0.25	1.0	KM1		0.03634	**1**	0.07014	0.07854	50910
0.5	0.05	KM1		0.09788	**1**	0.17831	0.05274	17322
0.5	0.1	KM1		0.12331	**0.99842**	0.21952	0.02964	13458
0.5	1.0	KM1		0.03624	**1**	0.06995	0.07877	51060
0.1	0.05	MS		**0.91472**	**1**	**0.95546**	**0.00053**	175.5
0.1	0.1	MS		**0.9503**	**1**	**0.97452**	**0.00022**	99
0.1	1.0	MS		0.65306	**1**	0.79012	**0.00157**	1020
0.25	0.05	MS		**0.93937**	**1**	**0.96874**	**0.00037**	121.5
0.25	0.1	MS		**0.89631**	**1**	**0.94532**	**0.00048**	219
0.25	1.0	MS		0.61538	**1**	0.7619	**0.00185**	1200
0.5	0.05	MS		**0.91393**	**1**	**0.95503**	**0.00054**	177
0.5	0.1	MS		**0.94048**	**1**	**0.96933**	**0.00026**	120
0.5	1.0	MS		0.64	**1**	0.78049	**0.00167**	1080

Tabelle A.8: Ergebnisse der Datenfusion für das Inject Lidar Szenario und der Methode Threshold.

Kombination				**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.25	0.05	HBO		0.24716	**1**	0.39635	0.01753	5761.5
0.25	0.1	HBO		0.20655	**1**	0.34238	0.01603	7272
0.25	1.0	HBO		0.08443	**1**	0.15572	0.03207	20820
0.5	0.05	HBO		0.21478	**1**	0.35362	0.02105	6915
0.5	0.1	HBO		0.19058	**1**	0.32014	0.01769	8040
0.5	1.0	HBO		0.08132	**1**	0.15041	0.03318	21690
1.0	0.05	HBO		0.20489	**1**	0.34009	0.02243	7369.5
1.0	0.1	HBO		0.18892	**1**	0.3178	0.01787	8127
1.0	1.0	HBO		0.08926	**1**	0.16389	0.02989	19590
0.25	0.05	IF		0.28571	0.00159	0.00315	**0.00002**	7.5
0.25	0.1	IF		**0.84615**	0.01743	0.03416	**0.00001**	6
0.25	1.0	IF		0	0	0	**0.00023**	150
0.5	0.05	IF		0	0	0	**0.00001**	4.5
0.5	0.1	IF		0.5	0.00951	0.01866	**0.00004**	18
0.5	1.0	IF		0	0	0	**0.00046**	300
1.0	0.05	IF		0.5	0.00237	0.00472	**0.00001**	4.5
1.0	0.1	IF		0.77778	0.02219	0.04314	**0.00003**	12
1.0	1.0	IF		0	0	0	**0.00014**	90
0.25	0.05	KM1		0.07785	**1**	0.14446	0.06815	22404
0.25	0.1	KM1		0.08998	**1**	0.1651	0.04221	19146
0.25	1.0	KM1		0.0349	**1**	0.06744	0.08179	53100
0.5	0.05	KM1		0.07303	**1**	0.13612	0.0731	24009
0.5	0.1	KM1		0.08658	**1**	0.15936	0.04395	19971
0.5	1.0	KM1		0.03342	**1**	0.06468	0.08496	55530
1.0	0.05	KM1		0.0728	**1**	0.13571	0.0736	24187.5
1.0	0.1	KM1		0.08503	**1**	0.15673	0.04479	20370
1.0	1.0	KM1		0.03421	**1**	0.06615	0.08271	54210
0.25	0.05	MS		0.24227	**1**	0.39004	0.018	5916
0.25	0.1	MS		0.20908	**1**	0.34585	0.01579	7161
0.25	1.0	MS		0.11594	**1**	0.20779	0.02255	14640
0.5	0.05	MS		0.21406	**1**	0.35263	0.02114	6945
0.5	0.1	MS		0.19899	**1**	0.33193	0.01677	7620
0.5	1.0	MS		0.12075	**1**	0.21549	0.02139	13980
1.0	0.05	MS		0.2069	**1**	0.34286	0.02215	7279.5
1.0	0.1	MS		0.19774	**1**	0.33019	0.01689	7680
1.0	1.0	MS		0.14382	**1**	0.25147	0.01744	11430

Tabelle A.9: Ergebnisse der Datenfusion für das Delay Manual Steer Szenario und der Methode Threshold.

Kombination				**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.1	0.05	HBO		0.42941	0.92314	0.58616	0.00714	2346
0.1	0.1	HBO		0.39086	0.90521	0.54597	0.00591	2679
0.1	1.0	HBO		0.15025	**0.95312**	0.25957	0.01615	10350
0.25	0.05	HBO		0.40397	**0.96519**	0.56956	0.00822	2700
0.25	0.1	HBO		0.37239	0.93038	0.53189	0.00655	2973
0.25	1.0	HBO		0.15025	**0.95312**	0.25957	0.01615	10350
0.5	0.05	HBO		0.36886	**0.98568**	0.53683	0.00968	3180
0.5	0.1	HBO		0.35465	**0.97183**	0.51967	0.00747	3390
0.5	1.0	HBO		0.14762	**0.96875**	0.2562	0.01676	10740
0.1	0.05	IF		0	0	0	**0.00001**	4.5
0.1	0.1	IF		0	0	0	**0.00002**	9
0.1	1.0	IF		0	0	0	**0.00033**	210
0.25	0.05	IF		0	0	0	**0.00002**	6
0.25	0.1	IF		0	0	0	**0.00008**	36
0.25	1.0	IF		0	0	0	**0.00014**	90
0.5	0.05	IF		0	0	0	**0.00003**	10.5
0.5	0.1	IF		0	0	0	**0.00006**	27
0.5	1.0	IF		0	0	0	**0.00014**	90
0.1	0.05	KM1		0.08221	0.92	0.15093	0.05983	19644
0.1	0.1	KM1		0.10357	**0.95893**	0.18694	0.03475	15762
0.1	1.0	KM1		0.03542	**1**	0.06841	0.08159	52290
0.25	0.05	KM1		0.08351	**0.95965**	0.15365	0.06081	19968
0.25	0.1	KM1		0.10365	**0.97627**	0.1874	0.03529	16008
0.25	1.0	KM1		0.03458	**0.98438**	0.06681	0.08233	52770
0.5	0.05	KM1		0.0818	**0.95704**	0.15072	0.06169	20254.5
0.5	0.1	KM1		0.10387	**0.98592**	0.18795	0.03594	16305
0.5	1.0	KM1		0.03288	**0.98438**	0.06364	0.08673	55590
0.1	0.05	MS		0.43824	0.9349	0.59675	0.00698	2292
0.1	0.1	MS		0.41942	0.92101	0.57637	0.00534	2421
0.1	1.0	MS		0.34078	**0.95312**	0.50206	0.00552	3540
0.25	0.05	MS		0.4065	**0.96994**	0.5729	0.00818	2685
0.25	0.1	MS		0.39496	0.94304	0.55675	0.00604	2739
0.25	1.0	MS		0.30808	**0.95312**	0.46565	0.00641	4110
0.5	0.05	MS		0.36966	**0.99045**	0.53838	0.0097	3184.5
0.5	0.1	MS		0.37485	**0.98435**	0.54294	0.00694	3147
0.5	1.0	MS		0.29665	**0.96875**	0.45421	0.00688	4410

Tabelle A.10: Ergebnisse der Datenfusion für das Reorder Camera Szenario und der Methode Threshold.

Kombination				**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.1	0.05	HBO		0	0	0	0.01306	4290
0.1	0.1	HBO		0.00053	0.00158	0.00079	0.01258	5709
0.1	1.0	HBO		0.00234	0.01562	0.00407	0.01988	12810
0.25	0.05	HBO		0	0	0	0.01784	5862
0.25	0.1	HBO		0	0	0	0.01414	6411
0.25	1.0	HBO		0	0	0	0.02386	15600
0.5	0.05	HBO		0	0	0	0.01982	6514.5
0.5	0.1	HBO		0	0	0	0.01743	7911
0.5	1.0	HBO		0	0	0	0.03021	19530
0.1	0.05	IF		0	0	0	**0.00002**	7.5
0.1	0.1	IF		0	0	0	**0.00001**	3
0.1	1.0	IF		0	0	0	**0.00023**	150
0.25	0.05	IF		0	0	0	**0.00001**	3
0.25	0.1	IF		0	0	0	**0.00003**	15
0.25	1.0	IF		0	0	0	**0.00023**	150
0.5	0.05	IF		0	0	0	**0.00002**	7.5
0.5	0.1	IF		0	0	0	**0.00005**	21
0.5	1.0	IF		0	0	0	**0.0007**	450
0.1	0.05	KM1		0.00022	0.00238	0.0004	0.06354	20871
0.1	0.1	KM1		0.00017	0.00158	0.0003	0.03956	17958
0.1	1.0	KM1		0.00055	0.01562	0.00106	0.08462	54540
0.25	0.05	KM1		0.0002	0.00238	0.00036	0.06946	22825.5
0.25	0.1	KM1		0.00079	0.00792	0.00143	0.04192	19011
0.25	1.0	KM1		0	0	0	0.08484	55470
0.5	0.05	KM1		0.00025	0.00317	0.00047	0.07266	23881.5
0.5	0.1	KM1		0.0003	0.00317	0.00055	0.04365	19812
0.5	1.0	KM1		0	0	0	0.08877	57390
0.1	0.05	MS		0	0	0	0.01288	4230
0.1	0.1	MS		0	0	0	0.01159	5262
0.1	1.0	MS		0	0	0	0.00991	6390
0.25	0.05	MS		0	0	0	0.01749	5748
0.25	0.1	MS		0	0	0	0.01316	5970
0.25	1.0	MS		0	0	0	0.01253	8190
0.5	0.05	MS		0	0	0	0.02159	7096.5
0.5	0.1	MS		0	0	0	0.01686	7653
0.5	1.0	MS		0	0	0	0.01429	9240

A.3.2 Tabellen: Mehrheitsfusion mit Fusion über die Zeit

Tabelle A.11: Ergebnisse der Datenfusion für das Baseline Szenario und der Methode Threshold Sliding Window.

Kombination				**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.0	0.05	HBO	WS: 0.50	0	0	0	**0.0001**	33
0.0	0.1	HBO	WS: 0.50	0	0	0	**0.00015**	66
0.0	1.0	HBO	WS: 0.50	0	0	0	**0.00285**	1830
0.0	0.05	IF	WS: 0.50	0	0	0	**0**	**0**
0.0	0.1	IF	WS: 0.50	0	0	0	**0**	**0**
0.0	1.0	IF	WS: 0.50	0	0	0	**0.00089**	570
0.0	0.05	KM1	WS: 0.50	0	0	0	0.00745	2442
0.0	0.1	KM1	WS: 0.50	0	0	0	0.01081	4884
0.0	1.0	KM1	WS: 0.50	0	0	0	0.08606	55320
0.0	0.05	MS	WS: 0.50	0	0	0	**0**	**0**
0.0	0.1	MS	WS: 0.50	0	0	0	**0**	**0**
0.0	1.0	MS	WS: 0.50	0	0	0	**0.00173**	1110

Tabelle A.12: Ergebnisse der Datenfusion für das Eliminate Auto Brake Szenario und der Methode Threshold Sliding Window.

Kombination				**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.1	0.05	HBO	WS: 0.50	**0.95365**	**1**	**0.97627**	**0.00028**	91.5
0.1	0.1	HBO	WS: 0.50	**0.91582**	**1**	**0.95606**	**0.00038**	174
0.1	1.0	HBO	WS: 0.50	0.19692	**1**	0.32905	0.01208	7830
0.25	0.05	HBO	WS: 0.50	**0.95437**	**1**	**0.97665**	**0.00027**	90
0.25	0.1	HBO	WS: 0.50	**0.91582**	**1**	**0.95606**	**0.00038**	174
0.25	1.0	HBO	WS: 0.50	0.19048	**1**	0.32	0.01259	8160
0.5	0.05	HBO	WS: 0.50	**0.9543**	**1**	**0.97662**	**0.00027**	90
0.5	0.1	HBO	WS: 0.50	**0.91594**	**1**	**0.95613**	**0.00038**	174
0.5	1.0	HBO	WS: 0.50	0.19453	**1**	0.3257	0.01226	7950
0.1	0.05	IF	WS: 0.50	0	0	0	**0**	**0**
0.1	0.1	IF	WS: 0.50	0	0	0	**0**	**0**
0.1	1.0	IF	WS: 0.50	0	0	0	**0.00014**	90
0.25	0.05	IF	WS: 0.50	0	0	0	**0**	**0**
0.25	0.1	IF	WS: 0.50	0	0	0	**0**	**0**
0.25	1.0	IF	WS: 0.50	0	0	0	**0.00023**	150
0.5	0.05	IF	WS: 0.50	0	0	0	**0**	**0**
0.5	0.1	IF	WS: 0.50	0	0	0	**0**	**0**
0.5	1.0	IF	WS: 0.50	0	0	0	**0.00019**	120
0.1	0.05	KM1	WS: 0.50	0.42185	**1**	0.59338	0.00786	2580
0.1	0.1	KM1	WS: 0.50	0.26874	**1**	0.42363	0.01134	5151
0.1	1.0	KM1	WS: 0.50	0.03534	**1**	0.06827	0.08085	52410
0.25	0.05	KM1	WS: 0.50	0.42213	**1**	0.59366	0.00785	2577
0.25	0.1	KM1	WS: 0.50	0.26874	**1**	0.42363	0.01134	5151
0.25	1.0	KM1	WS: 0.50	0.03548	**1**	0.06852	0.08053	52200
0.5	0.05	KM1	WS: 0.50	0.4216	**1**	0.59314	0.00785	2578.5
0.5	0.1	KM1	WS: 0.50	0.26874	**0.99842**	0.42349	0.01134	5151
0.5	1.0	KM1	WS: 0.50	0.0353	**1**	0.06819	0.08095	52470
0.1	0.05	MS	WS: 0.50	**0.99762**	**1**	**0.99881**	**0.00001**	4.5
0.1	0.1	MS	WS: 0.50	**1**	**1**	**1**	**0**	**0**
0.1	1.0	MS	WS: 0.50	0.62745	**1**	0.77108	**0.00176**	1140
0.25	0.05	MS	WS: 0.50	**0.99841**	**1**	**0.9992**	**0.00001**	3
0.25	0.1	MS	WS: 0.50	**1**	**1**	**1**	**0**	**0**
0.25	1.0	MS	WS: 0.50	0.59259	**1**	0.74419	**0.00204**	1320
0.5	0.05	MS	WS: 0.50	**0.99841**	**1**	**0.9992**	**0.00001**	3
0.5	0.1	MS	WS: 0.50	**1**	**1**	**1**	**0**	**0**
0.5	1.0	MS	WS: 0.50	0.62136	**1**	0.76647	**0.0018**	1170

Tabelle A.13: Ergebnisse der Datenfusion für das Inject Lidar Szenario und der Methode Threshold Sliding Window.

Kombination				**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.25	0.05	HBO	WS: 0.50	0.25641	**1**	0.40816	0.01669	5485.5
0.25	0.1	HBO	WS: 0.50	0.22408	**1**	0.36612	0.01445	6555
0.25	1.0	HBO	WS: 0.50	0.08312	**1**	0.15348	0.03262	21180
0.5	0.05	HBO	WS: 0.50	0.22676	**1**	0.36969	0.01964	6450
0.5	0.1	HBO	WS: 0.50	0.20929	**1**	0.34613	0.01574	7152
0.5	1.0	HBO	WS: 0.50	0.0802	**1**	0.14849	0.03369	22020
1.0	0.05	HBO	WS: 0.50	0.21075	**1**	0.34814	0.02164	7111.5
1.0	0.1	HBO	WS: 0.50	0.19893	**1**	0.33184	0.01676	7623
1.0	1.0	HBO	WS: 0.50	0.08743	**1**	0.1608	0.03058	20040
0.25	0.05	IF	WS: 0.50	0	0	0	**0**	**0**
0.25	0.1	IF	WS: 0.50	0	0	0	**0**	**0**
0.25	1.0	IF	WS: 0.50	0	0	0	**0.00028**	180
0.5	0.05	IF	WS: 0.50	0	0	0	**0**	**0**
0.5	0.1	IF	WS: 0.50	0	0	0	**0**	**0**
0.5	1.0	IF	WS: 0.50	0	0	0	**0.0006**	390
1.0	0.05	IF	WS: 0.50	0	0	0	**0**	**0**
1.0	0.1	IF	WS: 0.50	0	0	0	**0**	**0**
1.0	1.0	IF	WS: 0.50	0	0	0	**0.00014**	90
0.25	0.05	KM1	WS: 0.50	0.18659	**1**	0.3145	0.02508	8245.5
0.25	0.1	KM1	WS: 0.50	0.13771	**1**	0.24209	0.02613	11853
0.25	1.0	KM1	WS: 0.50	0.03406	**1**	0.06588	0.08386	54450
0.5	0.05	KM1	WS: 0.50	0.17222	**1**	0.29384	0.02768	9091.5
0.5	0.1	KM1	WS: 0.50	0.13078	**1**	0.2313	0.02769	12582
0.5	1.0	KM1	WS: 0.50	0.03244	**1**	0.06284	0.08762	57270
1.0	0.05	KM1	WS: 0.50	0.1615	**1**	0.27809	0.03	9859.5
1.0	0.1	KM1	WS: 0.50	0.12666	**1**	0.22484	0.0287	13053
1.0	1.0	KM1	WS: 0.50	0.03307	**1**	0.06403	0.08564	56130
0.25	0.05	MS	WS: 0.50	0.25135	**1**	0.40172	0.01714	5634
0.25	0.1	MS	WS: 0.50	0.22448	**1**	0.36665	0.01442	6540
0.25	1.0	MS	WS: 0.50	0.11532	**1**	0.20679	0.02269	14730
0.5	0.05	MS	WS: 0.50	0.22659	**1**	0.36947	0.01966	6456
0.5	0.1	MS	WS: 0.50	0.21217	**1**	0.35007	0.01547	7029
0.5	1.0	MS	WS: 0.50	0.11808	**1**	0.21122	0.02194	14340
1.0	0.05	MS	WS: 0.50	0.21202	**1**	0.34987	0.02148	7057.5
1.0	0.1	MS	WS: 0.50	0.20296	**1**	0.33743	0.01635	7434
1.0	1.0	MS	WS: 0.50	0.14254	**1**	0.24951	0.01762	11550

Tabelle A.14: Ergebnisse der Datenfusion für das Delay Manual Steer Szenario und der Methode Threshold Sliding Window.

Kombination				**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.1	0.05	HBO	WS: 0.50	0.40253	**0.99922**	0.57387	0.00864	2836.5
0.1	0.1	HBO	WS: 0.50	0.40541	**0.99526**	0.57613	0.00611	2772
0.1	1.0	HBO	WS: 0.50	0.1477	**0.95312**	0.25577	0.01648	10560
0.25	0.05	HBO	WS: 0.50	0.34928	**0.99921**	0.51762	0.01075	3529.5
0.25	0.1	HBO	WS: 0.50	0.36879	**0.99842**	0.53863	0.00714	3240
0.25	1.0	HBO	WS: 0.50	0.14699	**0.95312**	0.2547	0.01657	10620
0.5	0.05	HBO	WS: 0.50	0.33272	**1**	0.4993	0.01152	3781.5
0.5	0.1	HBO	WS: 0.50	0.33989	**1**	0.50734	0.00821	3723
0.5	1.0	HBO	WS: 0.50	0.14452	**0.96875**	0.25152	0.01718	11010
0.1	0.05	IF	WS: 0.50	0	0	0	**0**	**0**
0.1	0.1	IF	WS: 0.50	0	0	0	**0**	**0**
0.1	1.0	IF	WS: 0.50	0	0	0	**0.00047**	300
0.25	0.05	IF	WS: 0.50	0	0	0	**0**	**0**
0.25	0.1	IF	WS: 0.50	0	0	0	**0**	**0**
0.25	1.0	IF	WS: 0.50	0	0	0	**0.00014**	90
0.5	0.05	IF	WS: 0.50	0	0	0	**0**	**0**
0.5	0.1	IF	WS: 0.50	0	0	0	**0**	**0**
0.5	1.0	IF	WS: 0.50	0	0	0	**0.00014**	90
0.1	0.05	KM1	WS: 0.50	0.25188	**0.99765**	0.40221	0.01726	5667
0.1	0.1	KM1	WS: 0.50	0.17652	**1**	0.30007	0.01953	8859
0.1	1.0	KM1	WS: 0.50	0.03471	**1**	0.06709	0.08332	53400
0.25	0.05	KM1	WS: 0.50	0.23147	**0.99842**	0.37582	0.01914	6285
0.25	0.1	KM1	WS: 0.50	0.17551	**1**	0.29861	0.01963	8907
0.25	1.0	KM1	WS: 0.50	0.034	**0.98438**	0.06573	0.08379	53700
0.5	0.05	KM1	WS: 0.50	0.22721	**0.9992**	0.37023	0.01952	6408
0.5	0.1	KM1	WS: 0.50	0.17359	**1**	0.29583	0.02012	9126
0.5	1.0	KM1	WS: 0.50	0.0319	**0.98438**	0.06179	0.0895	57360
0.1	0.05	MS	WS: 0.50	0.40684	**0.99843**	0.57811	0.00848	2784
0.1	0.1	MS	WS: 0.50	0.40921	**0.99684**	0.58023	0.00602	2733
0.1	1.0	MS	WS: 0.50	0.32973	**0.95312**	0.48996	0.0058	3720
0.25	0.05	MS	WS: 0.50	0.34641	**0.99921**	0.51446	0.01089	3574.5
0.25	0.1	MS	WS: 0.50	0.37183	**0.99842**	0.54186	0.00705	3198
0.25	1.0	MS	WS: 0.50	0.2891	**0.95312**	0.44364	0.00702	4500
0.5	0.05	MS	WS: 0.50	0.33565	**1**	0.5026	0.01137	3732
0.5	0.1	MS	WS: 0.50	0.34392	**1**	0.51181	0.00806	3657
0.5	1.0	MS	WS: 0.50	0.28182	**0.96875**	0.43662	0.0074	4740

Tabelle A.15: Ergebnisse der Datenfusion für das Reorder Camera Szenario und der Methode Threshold Sliding Window.

Kombination				**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.1	0.05	HBO	WS: 0.50	0	0	0	0.01725	5665.5
0.1	0.1	HBO	WS: 0.50	0	0	0	0.01305	5925
0.1	1.0	HBO	WS: 0.50	0.00231	0.01562	0.00403	0.02006	12930
0.25	0.05	HBO	WS: 0.50	0	0	0	0.0178	5850
0.25	0.1	HBO	WS: 0.50	0	0	0	0.01317	5973
0.25	1.0	HBO	WS: 0.50	0	0	0	0.02432	15900
0.5	0.05	HBO	WS: 0.50	0	0	0	0.01978	6502.5
0.5	0.1	HBO	WS: 0.50	0	0	0	0.0169	7671
0.5	1.0	HBO	WS: 0.50	0	0	0	0.03058	19770
0.1	0.05	IF	WS: 0.50	0	0	0	**0**	**0**
0.1	0.1	IF	WS: 0.50	0	0	0	**0**	**0**
0.1	1.0	IF	WS: 0.50	0	0	0	**0.00023**	150
0.25	0.05	IF	WS: 0.50	0	0	0	**0**	**0**
0.25	0.1	IF	WS: 0.50	0	0	0	**0**	**0**
0.25	1.0	IF	WS: 0.50	0	0	0	**0.00028**	180
0.5	0.05	IF	WS: 0.50	0	0	0	**0**	**0**
0.5	0.1	IF	WS: 0.50	0	0	0	**0**	**0**
0.5	1.0	IF	WS: 0.50	0	0	0	**0.00093**	600
0.1	0.05	KM1	WS: 0.50	0	0	0	0.02417	7938
0.1	0.1	KM1	WS: 0.50	0	0	0	0.02432	11040
0.1	1.0	KM1	WS: 0.50	0.00054	0.01562	0.00104	0.08672	55890
0.25	0.05	KM1	WS: 0.50	0	0	0	0.02562	8418
0.25	0.1	KM1	WS: 0.50	0	0	0	0.02465	11178
0.25	1.0	KM1	WS: 0.50	0	0	0	0.08718	57000
0.5	0.05	KM1	WS: 0.50	0	0	0	0.02741	9009
0.5	0.1	KM1	WS: 0.50	0	0	0	0.02588	11748
0.5	1.0	KM1	WS: 0.50	0	0	0	0.09123	58980
0.1	0.05	MS	WS: 0.50	0	0	0	0.01692	5557.5
0.1	0.1	MS	WS: 0.50	0	0	0	0.01253	5685
0.1	1.0	MS	WS: 0.50	0	0	0	0.01005	6480
0.25	0.05	MS	WS: 0.50	0	0	0	0.01742	5724
0.25	0.1	MS	WS: 0.50	0	0	0	0.01261	5721
0.25	1.0	MS	WS: 0.50	0	0	0	0.01276	8340
0.5	0.05	MS	WS: 0.50	0	0	0	0.02114	6949.5
0.5	0.1	MS	WS: 0.50	0	0	0	0.01671	7587
0.5	1.0	MS	WS: 0.50	0	0	0	0.01439	9300

A.3.3 Tabellen: Dempster-Shafer

Tabelle A.16: Ergebnisse der Datenfusion für das Baseline Szenario und der Methode Dempster Shafer.

Kombination				**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.0	0.05	HBO	A:0.80; N:0.95	0	0	0	**0.00031**	102
0.0	0.1	HBO	A:0.80; N:0.95	0	0	0	**0.0005**	228
0.0	1.0	HBO	A:0.80; N:0.95	0	0	0	**0.00327**	2100
0.0	0.05	IF	A:0.70; N:0.95	0	0	0	**0.00002**	6
0.0	0.1	IF	A:0.70; N:0.95	0	0	0	**0.00001**	3
0.0	1.0	IF	A:0.70; N:0.95	0	0	0	**0.00079**	510
0.0	0.05	KM1	A:0.90; N:0.80	0	0	0	0.12282	40249.5
0.0	0.1	KM1	A:0.90; N:0.80	0	0	0	0.12097	54648
0.0	1.0	KM1	A:0.90; N:0.80	0	0	0	0.35007	225030
0.0	0.05	MS	A:0.80; N:0.60	0	0	0	0.01478	4842
0.0	0.1	MS	A:0.80; N:0.60	0	0	0	0.01612	7284
0.0	1.0	MS	A:0.80; N:0.60	0	0	0	0.02805	18030

Tabelle A.17: Ergebnisse der Datenfusion für das Eliminate Auto Brake Szenario und der Methode Dempster Shafer.

Kombination				**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.1	0.05	HBO	A:0.60; N:1.00	**0.98814**	**0.99602**	**0.99206**	**0.00007**	22.5
0.1	0.1	HBO	A:0.60; N:1.00	**0.9937**	**1**	**0.99684**	**0.00003**	12
0.1	1.0	HBO	A:0.60; N:0.95	0.20063	**1**	0.3342	0.0118	7650
0.25	0.05	HBO	A:0.60; N:1.00	**0.97741**	**1**	**0.98858**	**0.00013**	43.5
0.25	0.1	HBO	A:0.60; N:1.00	**0.9937**	**1**	**0.99684**	**0.00003**	12
0.25	1.0	HBO	A:0.60; N:0.95	0.19512	**1**	0.32653	0.01222	7920
0.5	0.05	HBO	A:0.60; N:1.00	**0.95649**	**1**	**0.97776**	**0.00026**	85.5
0.5	0.1	HBO	A:0.60; N:1.00	**0.9937**	**0.99842**	**0.99605**	**0.00003**	12
0.5	1.0	HBO	A:0.60; N:0.95	0.19814	**1**	0.33075	0.01199	7770
0.1	0.05	IF	A:0.70; N:0.95	0	0	0	**0.00001**	3
0.1	0.1	IF	A:0.70; N:0.95	0	0	0	**0.00001**	6
0.1	1.0	IF	A:0.70; N:0.95	0	0	0	**0.00019**	120
0.25	0.05	IF	A:0.70; N:0.95	0	0	0	**0.00002**	6
0.25	0.1	IF	A:0.70; N:0.95	0	0	0	**0.00002**	9
0.25	1.0	IF	A:0.70; N:0.95	0	0	0	**0.00028**	180
0.5	0.05	IF	A:0.70; N:0.95	0	0	0	**0.00001**	3
0.5	0.1	IF	A:0.70; N:0.95	0	0	0	**0.00002**	9
0.5	1.0	IF	A:0.70; N:0.95	0	0	0	**0.00014**	90
0.1	0.05	KM1	A:0.70; N:1.00	0.49623	**0.99681**	0.66261	0.0058	1905
0.1	0.1	KM1	A:0.70; N:1.00	0.34708	**1**	0.51531	0.00784	3561
0.1	1.0	KM1	A:0.70; N:1.00	0.18659	**1**	0.3145	0.01291	8370
0.25	0.05	KM1	A:0.70; N:1.00	0.49507	**1**	0.66227	0.00585	1920
0.25	0.1	KM1	A:0.70; N:1.00	0.34108	**1**	0.50867	0.00805	3657
0.25	1.0	KM1	A:0.70; N:1.00	0.18879	**1**	0.31762	0.01273	8250
0.5	0.05	KM1	A:0.70; N:1.00	0.48755	**1**	0.65551	0.00602	1975.5
0.5	0.1	KM1	A:0.70; N:1.00	0.34556	**0.99842**	0.51343	0.0079	3585
0.5	1.0	KM1	A:0.70; N:1.00	0.18497	**1**	0.3122	0.01305	8460
0.1	0.05	MS	A:0.90; N:1.00	**0.91472**	**1**	**0.95546**	**0.00053**	175.5
0.1	0.1	MS	A:0.80; N:1.00	**0.98134**	**1**	**0.99058**	**0.00008**	36
0.1	1.0	MS	A:0.80; N:1.00	**0.86486**	**1**	**0.92754**	**0.00046**	300
0.25	0.05	MS	A:0.90; N:1.00	**0.97362**	**1**	**0.98664**	**0.00016**	51
0.25	0.1	MS	A:0.80; N:1.00	**0.93068**	**1**	**0.96409**	**0.00031**	141
0.25	1.0	MS	A:0.80; N:1.00	**0.81013**	**1**	**0.8951**	**0.00069**	450
0.5	0.05	MS	A:0.80; N:1.00	**0.95503**	**1**	**0.977**	**0.00027**	88.5
0.5	0.1	MS	A:0.80; N:1.00	**0.98287**	**0.99842**	**0.99058**	**0.00007**	33
0.5	1.0	MS	A:0.80; N:1.00	**0.87671**	**1**	**0.93431**	**0.00042**	270

Tabelle A.18: Ergebnisse der Datenfusion für das Inject Lidar Szenario und der Methode Dempster Shafer.

Kombination				**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.25	0.05	HBO	A:0.90; N:1.00	0.25952	**1**	0.41209	0.01642	5397
0.25	0.1	HBO	A:0.90; N:1.00	0.21933	**1**	0.35975	0.01485	6738
0.25	1.0	HBO	A:0.60; N:1.00	0.73563	**1**	**0.84768**	**0.00106**	690
0.5	0.05	HBO	A:0.90; N:1.00	0.22386	**1**	0.36583	0.01997	6558
0.5	0.1	HBO	A:0.60; N:0.95	0.19058	**1**	0.32014	0.01769	8040
0.5	1.0	HBO	A:0.60; N:1.00	0.65306	**1**	0.79012	**0.00156**	1020
1.0	0.05	HBO	A:0.60; N:0.95	0.20489	**1**	0.34009	0.02243	7369.5
1.0	0.1	HBO	A:0.90; N:1.00	0.19961	**0.9794**	0.33163	0.01635	7434
1.0	1.0	HBO	A:0.60; N:1.00	0.66667	**1**	0.8	**0.00146**	960
0.25	0.05	IF	A:0.95; N:0.90	**0.90589**	**1**	**0.95062**	**0.0006**	196.5
0.25	0.1	IF	A:0.95; N:1.00	**0.99343**	**0.9588**	**0.97581**	**0.00003**	12
0.25	1.0	IF	A:0.95; N:0.70	0.01875	0.1875	0.03409	0.02902	18840
0.5	0.05	IF	A:0.95; N:0.90	**0.86846**	**1**	**0.9296**	**0.00087**	286.5
0.5	0.1	IF	A:0.90; N:0.95	**0.83087**	**0.98098**	**0.89971**	**0.00083**	378
0.5	1.0	IF	A:0.95; N:0.70	0.01559	0.20312	0.02895	0.03768	24630
1.0	0.05	IF	A:0.95; N:0.90	**0.87034**	**0.99684**	**0.92931**	**0.00086**	282
1.0	0.1	IF	A:0.95; N:1.00	**0.8695**	**0.97147**	**0.91766**	**0.00061**	276
1.0	1.0	IF	A:0.95; N:0.70	0.02433	0.29688	0.04497	0.03488	22860
0.25	0.05	KM1	A:0.70; N:1.00	0.20855	**0.99841**	0.34503	0.0218	7167
0.25	0.1	KM1	A:0.70; N:1.00	0.16953	**1**	0.28992	0.02044	9273
0.25	1.0	KM1	A:0.70; N:1.00	0.09816	**1**	0.17877	0.02717	17640
0.5	0.05	KM1	A:0.90; N:1.00	0.18841	**1**	0.31707	0.02481	8148
0.5	0.1	KM1	A:0.70; N:1.00	0.16048	**1**	0.27657	0.02179	9903
0.5	1.0	KM1	A:0.70; N:1.00	0.08964	**1**	0.16452	0.02983	19500
1.0	0.05	KM1	A:0.70; N:1.00	0.17981	**0.99605**	0.30463	0.02626	8628
1.0	0.1	KM1	A:0.70; N:1.00	0.1563	**1**	0.27035	0.02247	10218
1.0	1.0	KM1	A:0.70; N:1.00	0.09302	**1**	0.17021	0.02856	18720
0.25	0.05	MS	A:0.80; N:1.00	0.24605	**1**	0.39493	0.01763	5796
0.25	0.1	MS	A:0.80; N:1.00	0.21239	**1**	0.35036	0.01548	7020
0.25	1.0	MS	A:0.80; N:1.00	0.11808	**1**	0.21122	0.02209	14340
0.5	0.05	MS	A:0.80; N:1.00	0.21726	**1**	0.35697	0.02075	6814.5
0.5	0.1	MS	A:0.80; N:1.00	0.20096	**1**	0.33466	0.01656	7527
0.5	1.0	MS	A:0.80; N:1.00	0.12451	**1**	0.22145	0.02065	13500
1.0	0.05	MS	A:0.80; N:1.00	0.20908	**1**	0.34585	0.02186	7183.5
1.0	0.1	MS	A:0.80; N:1.00	0.20051	**1**	0.33404	0.0166	7548
1.0	1.0	MS	A:0.80; N:1.00	0.15059	**1**	0.26176	0.01652	10830

Tabelle A.19: Ergebnisse der Datenfusion für das Delay Manual Steer Szenario und der Methode Dempster Shafer.

Kombination				**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.1	0.05	HBO	A:0.60; N:1.00	0.54671	0.84	0.66234	**0.00406**	1332
0.1	0.1	HBO	A:0.60; N:1.00	0.51512	0.80727	0.62892	**0.00318**	1443
0.1	1.0	HBO	A:0.60; N:0.90	0.14988	**0.95312**	0.25902	0.0162	10380
0.25	0.05	HBO	A:0.60; N:1.00	0.51479	0.92247	0.66081	0.00502	1648.5
0.25	0.1	HBO	A:0.60; N:1.00	0.49907	0.85285	0.62967	**0.00358**	1623
0.25	1.0	HBO	A:0.60; N:0.90	0.15025	**0.95312**	0.25957	0.01615	10350
0.5	0.05	HBO	A:0.60; N:1.00	0.4515	0.93317	0.60856	0.00651	2137.5
0.5	0.1	HBO	A:0.60; N:1.00	0.47276	0.92332	0.62533	**0.00435**	1974
0.5	1.0	HBO	A:0.60; N:0.90	0.14762	**0.96875**	0.2562	0.01676	10740
0.1	0.05	IF	A:0.90; N:0.70	0.12069	0.03843	0.0583	**0.00163**	535.5
0.1	0.1	IF	A:0.70; N:0.95	0	0	0	**0.00002**	9
0.1	1.0	IF	A:0.70; N:0.95	0	0	0	**0.00042**	270
0.25	0.05	IF	A:0.90; N:0.70	0.16216	0.05222	0.07899	**0.00156**	511.5
0.25	0.1	IF	A:0.70; N:0.95	0	0	0	**0.00008**	36
0.25	1.0	IF	A:0.70; N:0.95	0	0	0	**0.00023**	150
0.5	0.05	IF	A:0.80; N:0.80	0.18395	0.04375	0.07069	**0.00111**	366
0.5	0.1	IF	A:0.70; N:0.95	0	0	0	**0.00006**	27
0.5	1.0	IF	A:0.70; N:0.95	0	0	0	**0.00023**	150
0.1	0.05	KM1	A:0.70; N:1.00	0.31763	0.92	0.47222	0.01151	3780
0.1	0.1	KM1	A:0.70; N:1.00	0.24359	0.93049	0.3861	0.0121	5487
0.1	1.0	KM1	A:0.70; N:1.00	0.12449	**0.95312**	0.22022	0.02008	12870
0.25	0.05	KM1	A:0.70; N:1.00	0.30655	**0.95965**	0.46466	0.01254	4116
0.25	0.1	KM1	A:0.70; N:1.00	0.24114	0.94778	0.38447	0.01247	5655
0.25	1.0	KM1	A:0.70; N:1.00	0.12224	**0.95312**	0.2167	0.0205	13140
0.5	0.05	KM1	A:0.70; N:1.00	0.29213	**0.95704**	0.44763	0.01332	4372.5
0.5	0.1	KM1	A:0.70; N:1.00	0.23606	**0.96088**	0.37901	0.01314	5961
0.5	1.0	KM1	A:0.70; N:1.00	0.10535	**0.98438**	0.19033	0.02504	16050
0.1	0.05	MS	A:0.80; N:1.00	0.52444	0.86667	0.65346	**0.00458**	1503
0.1	0.1	MS	A:0.80; N:1.00	0.49352	0.84202	0.6223	**0.00362**	1641
0.1	1.0	MS	A:0.80; N:1.00	0.45536	0.79688	0.57955	**0.00286**	1830
0.25	0.05	MS	A:0.80; N:1.00	0.49478	0.9375	0.64772	0.00553	1815
0.25	0.1	MS	A:0.80; N:1.00	0.47521	0.87975	0.61709	**0.00406**	1842
0.25	1.0	MS	A:0.80; N:1.00	0.41135	0.90625	0.56585	**0.00389**	2490
0.5	0.05	MS	A:0.80; N:1.00	0.43264	**0.95545**	0.59559	0.0072	2362.5
0.5	0.1	MS	A:0.80; N:1.00	0.45104	**0.95149**	0.61198	**0.00489**	2220
0.5	1.0	MS	A:0.80; N:1.00	0.35329	0.92188	0.51082	0.00506	3240

Tabelle A.20: Ergebnisse der Datenfusion für das Reorder Camera Szenario und der Methode Dempster Shafer.

Kombination				**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.1	0.05	HBO	A:0.90; N:0.60	0.01216	0.03251	0.0177	0.01521	4996.5
0.1	0.1	HBO	A:0.95; N:0.90	0.00091	0.00317	0.00142	0.01449	6579
0.1	1.0	HBO	A:0.70; N:0.80	0.07958	0.9375	0.1467	0.0323	20820
0.25	0.05	HBO	A:0.90; N:0.60	0.01009	0.0373	0.01588	0.02105	6916.5
0.25	0.1	HBO	A:0.80; N:0.60	0.0007	0.00317	0.00114	0.01898	8607
0.25	1.0	HBO	A:0.70; N:0.80	0.06667	**0.95312**	0.12462	0.03919	25620
0.5	0.05	HBO	A:0.90; N:0.60	0.00827	0.03889	0.01364	0.02681	8811
0.5	0.1	HBO	A:0.80; N:0.95	0	0	0	0.01868	8478
0.5	1.0	HBO	A:0.70; N:0.80	0.05561	**0.96875**	0.10517	0.04886	31590
0.1	0.05	IF	A:0.95; N:0.60	0.00583	0.00317	0.00411	**0.00311**	1023
0.1	0.1	IF	A:0.90; N:0.60	0.00426	0.00317	0.00363	**0.00309**	1404
0.1	1.0	IF	A:0.70; N:0.95	0	0	0	**0.00028**	180
0.25	0.05	IF	A:0.95; N:0.70	0.00585	0.00317	0.00412	**0.0031**	1020
0.25	0.1	IF	A:0.90; N:0.60	0.00426	0.00475	0.00449	**0.00464**	2106
0.25	1.0	IF	A:0.70; N:0.95	0	0	0	**0.00032**	210
0.5	0.05	IF	A:0.95; N:0.60	0.00086	0.00159	0.00111	0.01065	3501
0.5	0.1	IF	A:0.90; N:0.70	0.001	0.00158	0.00123	0.00661	3000
0.5	1.0	IF	A:0.70; N:0.95	0	0	0	**0.00107**	690
0.1	0.05	KM1	A:0.95; N:0.60	0.00273	0.1245	0.00534	0.26199	86050.5
0.1	0.1	KM1	A:0.95; N:0.70	0.01152	0.53407	0.02256	0.19111	86742
0.1	1.0	KM1	A:0.95; N:0.60	0.0033	0.60938	0.00657	0.54799	353190
0.25	0.05	KM1	A:0.95; N:0.70	0.00483	0.16032	0.00937	0.19015	62482.5
0.25	0.1	KM1	A:0.95; N:0.70	0.01565	0.75119	0.03066	0.19723	89451
0.25	1.0	KM1	A:0.95; N:0.60	0.0047	0.89062	0.00935	0.55387	362130
0.5	0.05	KM1	A:0.95; N:0.70	0.00844	0.29206	0.01641	0.19722	64822.5
0.5	0.1	KM1	A:0.95; N:0.70	0.01734	0.8542	0.03398	0.20191	91653
0.5	1.0	KM1	A:0.95; N:0.60	0.00489	0.9375	0.00973	0.5665	366240
0.1	0.05	MS	A:0.80; N:0.60	0.14992	0.83426	0.25417	0.02724	8947.5
0.1	0.1	MS	A:0.80; N:0.60	0.10605	0.89699	0.18968	0.03153	14313
0.1	1.0	MS	A:0.80; N:0.60	0.03138	0.9375	0.06073	0.0862	55560
0.25	0.05	MS	A:0.80; N:0.60	0.13383	0.89603	0.23288	0.03336	10960.5
0.25	0.1	MS	A:0.80; N:0.60	0.09803	0.93027	0.17737	0.03573	16203
0.25	1.0	MS	A:0.80; N:0.60	0.02981	**0.95312**	0.05782	0.09108	59550
0.5	0.05	MS	A:0.80; N:0.60	0.10997	0.91825	0.19642	0.04273	14046
0.5	0.1	MS	A:0.80; N:0.60	0.08318	0.91442	0.15248	0.04203	19080
0.5	1.0	MS	A:0.80; N:0.60	0.02634	**0.96875**	0.05128	0.10636	68760

A.3.4 Tabellen: Dempster-Shafer mit Fusion über die Zeit

Tabelle A.21: Ergebnisse der Datenfusion für das Baseline Szenario und der Methode Dempster Shafer Sliding Window.

Kombination				**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.0	0.05	HBO	A:0.80; N:0.95; WS: 0.5	0	0	0	**0.0001**	33
0.0	0.1	HBO	A:0.80; N:0.95; WS: 0.5	0	0	0	**0.00015**	66
0.0	1.0	HBO	A:0.80; N:0.95; WS: 0.5	0	0	0	**0.00331**	2130
0.0	0.05	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0**	**0**
0.0	0.1	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0**	**0**
0.0	1.0	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0.00103**	660
0.0	0.05	KM1	A:0.90; N:0.80; WS: 0.5	0	0	0	0.02434	7977
0.0	0.1	KM1	A:0.90; N:0.80; WS: 0.5	0	0	0	0.03105	14025
0.0	1.0	KM1	A:0.90; N:0.80; WS: 0.5	0	0	0	0.35217	226380
0.0	0.05	MS	A:0.80; N:0.60; WS: 0.5	0	0	0	**0.00007**	22.5
0.0	0.1	MS	A:0.80; N:0.60; WS: 0.5	0	0	0	0.01347	6084
0.0	1.0	MS	A:0.80; N:0.60; WS: 0.5	0	0	0	0.02917	18750

Tabelle A.22: Ergebnisse der Datenfusion für das Eliminate Auto Brake Szenario und der Methode Dempster Shafer Sliding Window.

Kombination				Ergebnis				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.1	0.05	HBO	A:0.60; N:1.00; WS: 0.5	**0.99841**	**0.9992**	**0.99881**	**0.00001**	3
0.1	0.1	HBO	A:0.60; N:1.00; WS: 0.5	**1**	**1**	**1**	**0**	**0**
0.1	1.0	HBO	A:0.60; N:0.95; WS: 0.5	0.19692	**1**	0.32905	0.01208	7830
0.25	0.05	HBO	A:0.60; N:1.00; WS: 0.5	**0.9992**	**1**	**0.9996**	**0**	1.5
0.25	0.1	HBO	A:0.95; N:1.00; WS: 0.5	**1**	**1**	**1**	**0**	**0**
0.25	1.0	HBO	A:0.60; N:0.95; WS: 0.5	0.19048	**1**	0.32	0.01259	8160
0.5	0.05	HBO	A:0.60; N:1.00; WS: 0.5	**0.9992**	**1**	**0.9996**	**0**	1.5
0.5	0.1	HBO	A:0.60; N:1.00; WS: 0.5	**1**	**0.99842**	**0.99921**	**0**	**0**
0.5	1.0	HBO	A:0.60; N:0.95; WS: 0.5	0.19453	**1**	0.3257	0.01226	7950
0.1	0.05	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0**	**0**
0.1	0.1	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0**	**0**
0.1	1.0	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0.00019**	120
0.25	0.05	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0**	**0**
0.25	0.1	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0**	**0**
0.25	1.0	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0.00032**	210
0.5	0.05	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0**	**0**
0.5	0.1	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0**	**0**
0.5	1.0	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0.00019**	120
0.1	0.05	KM1	A:0.70; N:1.00; WS: 0.5	**0.99682**	**0.9992**	**0.99801**	**0.00002**	6
0.1	0.1	KM1	A:0.70; N:1.00; WS: 0.5	**0.99684**	**1**	**0.99842**	**0.00001**	6
0.1	1.0	KM1	A:0.70; N:1.00; WS: 0.5	0.17251	**1**	0.29425	0.01421	9210
0.25	0.05	KM1	A:0.70; N:1.00; WS: 0.5	**0.99841**	**1**	**0.9992**	**0.00001**	3
0.25	0.1	KM1	A:0.70; N:1.00; WS: 0.5	**0.99684**	**1**	**0.99842**	**0.00001**	6
0.25	1.0	KM1	A:0.70; N:1.00; WS: 0.5	0.1768	**1**	0.30047	0.01379	8940
0.5	0.05	KM1	A:0.70; N:1.00; WS: 0.5	**0.99761**	**1**	**0.9988**	**0.00001**	4.5
0.5	0.1	KM1	A:0.70; N:1.00; WS: 0.5	**0.99684**	**0.99842**	**0.99763**	**0.00001**	6
0.5	1.0	KM1	A:0.70; N:1.00; WS: 0.5	0.17158	**1**	0.29291	0.0143	9270
0.1	0.05	MS	A:0.90; N:1.00; WS: 0.5	**0.99762**	**1**	**0.99881**	**0.00001**	4.5
0.1	0.1	MS	A:0.90; N:1.00; WS: 0.5	**1**	**1**	**1**	**0**	**0**
0.1	1.0	MS	A:0.80; N:1.00; WS: 0.5	**0.82051**	**1**	**0.90141**	**0.00065**	420
0.25	0.05	MS	A:0.90; N:1.00; WS: 0.5	**0.9992**	**1**	**0.9996**	**0**	1.5
0.25	0.1	MS	A:0.90; N:1.00; WS: 0.5	**1**	**1**	**1**	**0**	**0**
0.25	1.0	MS	A:0.80; N:1.00; WS: 0.5	0.77108	**1**	**0.87075**	**0.00088**	570
0.5	0.05	MS	A:0.80; N:1.00; WS: 0.5	**0.9992**	**1**	**0.9996**	**0**	1.5
0.5	0.1	MS	A:0.60; N:0.90; WS: 0.5	**1**	**1**	**1**	**0**	**0**
0.5	1.0	MS	A:0.80; N:1.00; WS: 0.5	**0.84211**	**1**	**0.91429**	**0.00056**	360

Tabelle A.23: Ergebnisse der Datenfusion für das Inject Lidar Szenario und der Methode Dempster Shafer Sliding Window.

Kombination				Ergebnis				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.25	0.05	HBO	A:0.95; N:1.00; WS: 0.5	0.26419	**1**	0.41796	0.01602	5268
0.25	0.1	HBO	A:0.95; N:1.00; WS: 0.5	0.23164	**1**	0.37615	0.01384	6279
0.25	1.0	HBO	A:0.60; N:1.00; WS: 0.5	0.71111	**1**	**0.83117**	**0.0012**	780
0.5	0.05	HBO	A:0.90; N:1.00; WS: 0.5	0.23291	**1**	0.37783	0.01897	6229.5
0.5	0.1	HBO	A:0.70; N:0.95; WS: 0.5	0.20929	**1**	0.34613	0.01574	7152
0.5	1.0	HBO	A:0.60; N:1.00; WS: 0.5	0.64646	**1**	0.78528	**0.00161**	1050
1.0	0.05	HBO	A:0.80; N:0.95; WS: 0.5	0.21075	**1**	0.34814	0.02164	7111.5
1.0	0.1	HBO	A:0.90; N:1.00; WS: 0.5	0.20709	**1**	0.34312	0.01594	7248
1.0	1.0	HBO	A:0.60; N:1.00; WS: 0.5	0.62136	**1**	0.76647	**0.00179**	1170
0.25	0.05	IF	A:0.60; N:0.60; WS: 0.5	**0.98585**	**0.99445**	**0.99013**	**0.00008**	27
0.25	0.1	IF	A:0.95; N:1.00; WS: 0.5	**1**	**0.99842**	**0.99921**	**0**	**0**
0.25	1.0	IF	A:0.95; N:0.70; WS: 0.5	0.01832	0.1875	0.03338	0.02971	19290
0.5	0.05	IF	A:0.60; N:0.70; WS: 0.5	**1**	**0.98255**	**0.9912**	**0**	**0**
0.5	0.1	IF	A:0.90; N:0.90; WS: 0.5	**0.94039**	**1**	**0.96928**	**0.00026**	120
0.5	1.0	IF	A:0.95; N:0.70; WS: 0.5	0.01517	0.20312	0.02823	0.03874	25320
1.0	0.05	IF	A:0.90; N:0.80; WS: 0.5	**0.96568**	**1**	**0.98254**	**0.00021**	67.5
1.0	0.1	IF	A:0.90; N:0.90; WS: 0.5	**0.93899**	**1**	**0.96853**	**0.00027**	123
1.0	1.0	IF	A:0.95; N:0.70; WS: 0.5	0.02396	0.29688	0.04434	0.03543	23220
0.25	0.05	KM1	A:0.70; N:1.00; WS: 0.5	0.24887	**1**	0.39855	0.01737	5709
0.25	0.1	KM1	A:0.70; N:1.00; WS: 0.5	0.21684	**1**	0.3564	0.01507	6837
0.25	1.0	KM1	A:0.70; N:1.00; WS: 0.5	0.0944	**1**	0.17251	0.02837	18420
0.5	0.05	KM1	A:0.70; N:1.00; WS: 0.5	0.22631	**1**	0.36909	0.01969	6466.5
0.5	0.1	KM1	A:0.70; N:1.00; WS: 0.5	0.20474	**1**	0.33989	0.01618	7353
0.5	1.0	KM1	A:0.70; N:1.00; WS: 0.5	0.08625	**1**	0.15881	0.03112	20340
1.0	0.05	KM1	A:0.70; N:1.00; WS: 0.5	0.20936	**1**	0.34623	0.02182	7171.5
1.0	0.1	KM1	A:0.70; N:1.00; WS: 0.5	0.19737	**1**	0.32968	0.01693	7698
1.0	1.0	KM1	A:0.70; N:1.00; WS: 0.5	0.08864	**1**	0.16285	0.03012	19740
0.25	0.05	MS	A:0.80; N:1.00; WS: 0.5	0.25357	**1**	0.40456	0.01694	5568
0.25	0.1	MS	A:0.80; N:1.00; WS: 0.5	0.22633	**1**	0.36911	0.01427	6471
0.25	1.0	MS	A:0.80; N:1.00; WS: 0.5	0.11743	**1**	0.21018	0.02223	14430
0.5	0.05	MS	A:0.80; N:1.00; WS: 0.5	0.22762	**1**	0.37083	0.01954	6418.5
0.5	0.1	MS	A:0.80; N:1.00; WS: 0.5	0.21339	**1**	0.35173	0.01536	6978
0.5	1.0	MS	A:0.80; N:1.00; WS: 0.5	0.1219	**1**	0.21732	0.02116	13830
1.0	0.05	MS	A:0.80; N:1.00; WS: 0.5	0.2127	**1**	0.35079	0.02139	7029
1.0	0.1	MS	A:0.80; N:1.00; WS: 0.5	0.20441	**1**	0.33943	0.0162	7368
1.0	1.0	MS	A:0.80; N:1.00; WS: 0.5	0.14918	**1**	0.25963	0.01671	10950

Tabelle A.24: Ergebnisse der Datenfusion für das Delay Manual Steer Szenario und der Methode Dempster Shafer Sliding Window.

Kombination				Ergebnis				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.1	0.05	HBO	A:0.60; N:1.00; WS: 0.5	0.75671	**0.99529**	**0.85976**	**0.00186**	612
0.1	0.1	HBO	A:0.60; N:1.00; WS: 0.5	0.63957	0.94471	0.76276	**0.00223**	1011
0.1	1.0	HBO	A:0.60; N:0.90; WS: 0.5	0.1477	**0.95312**	0.25577	0.01648	10560
0.25	0.05	HBO	A:0.60; N:1.00; WS: 0.5	0.61723	**0.99763**	0.76262	**0.00357**	1173
0.25	0.1	HBO	A:0.60; N:1.00; WS: 0.5	0.60038	**0.98892**	0.74716	**0.00275**	1248
0.25	1.0	HBO	A:0.60; N:0.90; WS: 0.5	0.14699	**0.95312**	0.2547	0.01657	10620
0.5	0.05	HBO	A:0.60; N:1.00; WS: 0.5	0.4805	**1**	0.64911	0.00621	2038.5
0.5	0.1	HBO	A:0.95; N:1.00; WS: 0.5	0.49689	**1**	0.6639	**0.00428**	1941
0.5	1.0	HBO	A:0.60; N:0.90; WS: 0.5	0.14452	**0.96875**	0.25152	0.01718	11010
0.1	0.05	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0**	**0**
0.1	0.1	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0**	**0**
0.1	1.0	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0.00056**	360
0.25	0.05	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0**	**0**
0.25	0.1	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0**	**0**
0.25	1.0	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0.00023**	150
0.5	0.05	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0**	**0**
0.5	0.1	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0**	**0**
0.5	1.0	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0.00023**	150
0.1	0.05	KM1	A:0.70; N:1.00; WS: 0.5	0.41192	**0.99765**	0.58309	0.0083	2724
0.1	0.1	KM1	A:0.70; N:1.00; WS: 0.5	0.35467	**0.98894**	0.5221	0.00753	3417
0.1	1.0	KM1	A:0.70; N:1.00; WS: 0.5	0.11891	**0.95312**	0.21144	0.02116	13560
0.25	0.05	KM1	A:0.70; N:1.00; WS: 0.5	0.35036	**0.99842**	0.5187	0.01069	3510
0.25	0.1	KM1	A:0.70; N:1.00; WS: 0.5	0.35406	**1**	0.52296	0.00762	3459
0.25	1.0	KM1	A:0.70; N:1.00; WS: 0.5	0.11868	**0.95312**	0.21107	0.0212	13590
0.5	0.05	KM1	A:0.70; N:1.00; WS: 0.5	0.3443	**0.9992**	0.51213	0.01093	3588
0.5	0.1	KM1	A:0.95; N:1.00; WS: 0.5	0.34375	**0.99844**	0.51142	0.00805	3654
0.5	1.0	KM1	A:0.70; N:1.00; WS: 0.5	0.09844	**0.98438**	0.17898	0.02701	17310
0.1	0.05	MS	A:0.80; N:1.00; WS: 0.5	0.635	**0.99608**	0.77557	**0.00333**	1095
0.1	0.1	MS	A:0.80; N:1.00; WS: 0.5	0.57263	**0.96524**	0.71882	**0.00302**	1368
0.1	1.0	MS	A:0.80; N:1.00; WS: 0.5	0.4322	0.79688	0.56044	**0.00314**	2010
0.25	0.05	MS	A:0.80; N:1.00; WS: 0.5	0.53226	**0.99842**	0.69436	0.00507	1663.5
0.25	0.1	MS	A:0.80; N:1.00; WS: 0.5	0.53322	**0.99051**	0.69324	**0.00362**	1644
0.25	1.0	MS	A:0.80; N:1.00; WS: 0.5	0.37662	0.90625	0.53211	**0.00449**	2880
0.5	0.05	MS	A:0.80; N:1.00; WS: 0.5	0.41858	**1**	0.59014	0.00798	2619
0.5	0.1	MS	A:0.80; N:1.00; WS: 0.5	0.43117	**1**	0.60255	0.00557	2529
0.5	1.0	MS	A:0.80; N:1.00; WS: 0.5	0.33146	0.92188	0.4876	0.00557	3570

Tabelle A.25: Ergebnisse der Datenfusion für das Reorder Camera Szenario und der Methode Dempster Shafer Sliding Window.

Kombination				**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.1	0.05	HBO	A:0.80; N:0.95; WS: 0.5	0	0	0	0.01726	5670
0.1	0.1	HBO	A:0.80; N:0.95; WS: 0.5	0	0	0	0.01305	5925
0.1	1.0	HBO	A:0.70; N:0.80; WS: 0.5	0.07833	0.9375	0.14458	0.03286	21180
0.25	0.05	HBO	A:0.80; N:0.95; WS: 0.5	0	0	0	0.0178	5850
0.25	0.1	HBO	A:0.80; N:0.95; WS: 0.5	0	0	0	0.01317	5973
0.25	1.0	HBO	A:0.70; N:0.80; WS: 0.5	0.06455	**0.95312**	0.12091	0.04056	26520
0.5	0.05	HBO	A:0.80; N:0.95; WS: 0.5	0	0	0	0.02246	7383
0.5	0.1	HBO	A:0.80; N:0.95; WS: 0.5	0	0	0	0.01745	7920
0.5	1.0	HBO	A:0.70; N:0.80; WS: 0.5	0.05458	**0.96875**	0.10333	0.04984	32220
0.1	0.05	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0**	**0**
0.1	0.1	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0**	**0**
0.1	1.0	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0.00028**	180
0.25	0.05	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0**	**0**
0.25	0.1	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0**	**0**
0.25	1.0	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0.00037**	240
0.5	0.05	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0**	**0**
0.5	0.1	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0.00264**	1200
0.5	1.0	IF	A:0.70; N:0.95; WS: 0.5	0	0	0	**0.0013**	840
0.1	0.05	KM1	A:0.90; N:0.80; WS: 0.5	0	0	0	0.04269	14022
0.1	0.1	KM1	A:0.95; N:0.70; WS: 0.5	0.0292	0.59113	0.05564	0.08198	37209
0.1	1.0	KM1	A:0.95; N:0.60; WS: 0.5	0.00327	0.60938	0.0065	0.55409	357120
0.25	0.05	KM1	A:0.90; N:0.80; WS: 0.5	0	0	0	0.04339	14259
0.25	0.1	KM1	A:0.95; N:0.70; WS: 0.5	0.04282	0.94612	0.08194	0.08827	40032
0.25	1.0	KM1	A:0.95; N:0.60; WS: 0.5	0.00462	0.89062	0.00919	0.56337	368340
0.5	0.05	KM1	A:0.95; N:0.70; WS: 0.5	0.00134	0.02222	0.00253	0.09525	31305
0.5	0.1	KM1	A:0.95; N:0.70; WS: 0.5	0.04362	**1**	0.0836	0.09143	41502
0.5	1.0	KM1	A:0.95; N:0.60; WS: 0.5	0.00482	0.9375	0.0096	0.57457	371460
0.1	0.05	MS	A:0.80; N:0.60; WS: 0.5	0.24611	**0.99128**	0.39432	0.01749	5743.5
0.1	0.1	MS	A:0.80; N:0.60; WS: 0.5	0.13421	**0.99049**	0.23638	0.02665	12096
0.1	1.0	MS	A:0.80; N:0.60; WS: 0.5	0.03071	0.9375	0.05946	0.08816	56820
0.25	0.05	MS	A:0.80; N:0.60; WS: 0.5	0.23853	**0.99444**	0.38477	0.01826	6000
0.25	0.1	MS	A:0.80; N:0.60; WS: 0.5	0.12918	**0.99842**	0.22876	0.02809	12741
0.25	1.0	MS	A:0.80; N:0.60; WS: 0.5	0.02895	**0.95312**	0.0562	0.09388	61380
0.5	0.05	MS	A:0.80; N:0.60; WS: 0.5	0.16189	**0.99603**	0.27852	0.02965	9745.5
0.5	0.1	MS	A:0.80; N:0.60; WS: 0.5	0.1042	**0.99842**	0.18871	0.03579	16248
0.5	1.0	MS	A:0.80; N:0.60; WS: 0.5	0.02575	**0.96875**	0.05016	0.10886	70380

A.3.5 Tabellen: Multi-Level Dempster-Shafer

Tabelle A.26: Ergebnisse der Datenfusion für das Baseline Szenario und der Methode Multi-Level Dempster Shafer.

Kombination				**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.0	0.05		a:0	0	0	0	**0**	**0**
0.0	0.05		a:5	0	0	0	**0**	**0**
0.0	0.05		a:-5	0	0	0	**0**	**0**
0.0	0.1		a:0	0	0	0	**0**	**0**
0.0	0.1		a:5	0	0	0	**0**	**0**
0.0	0.1		a:-5	0	0	0	**0**	**0**
0.0	1.0		a:0	0	0	0	**0.00009**	60
0.0	1.0		a:5	0	0	0	**0.00009**	60
0.0	1.0		a:-5	0	0	0	**0.00009**	60

Tabelle A.27: Ergebnisse der Datenfusion für das Eliminate Auto Brake Szenario und der Methode Multi-Level Dempster Shafer.

Kombination				**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.1	0.05		a:0	**0.99841**	**0.9992**	**0.99881**	**0.00001**	3
0.1	0.05		a:5	**0.99841**	**0.9992**	**0.99881**	**0.00001**	3
0.1	0.05		a:-5	**0.99841**	**0.9992**	**0.99881**	**0.00001**	3
0.1	0.1		a:0	**1**	**1**	**1**	**0**	**0**
0.1	0.1		a:5	**1**	**1**	**1**	**0**	**0**
0.1	0.1		a:-5	**1**	**1**	**1**	**0**	**0**
0.1	1.0		a:0	**0.87671**	**1**	**0.93431**	**0.00042**	270
0.1	1.0		a:5	**0.87671**	**1**	**0.93431**	**0.00042**	270
0.1	1.0		a:-5	**0.87671**	**1**	**0.93431**	**0.00042**	270
0.25	0.05		a:0	**0.9992**	**1**	**0.9996**	**0**	1.5
0.25	0.05		a:5	**0.9992**	**1**	**0.9996**	**0**	1.5
0.25	0.05		a:-5	**0.9992**	**1**	**0.9996**	**0**	1.5
0.25	0.1		a:0	**1**	**1**	**1**	**0**	**0**
0.25	0.1		a:5	**1**	**1**	**1**	**0**	**0**
0.25	0.1		a:-5	**1**	**1**	**1**	**0**	**0**
0.25	1.0		a:0	**0.83117**	**1**	**0.9078**	**0.00061**	390
0.25	1.0		a:5	**0.83117**	**1**	**0.9078**	**0.00061**	390
0.25	1.0		a:-5	**0.83117**	**1**	**0.9078**	**0.00061**	390
0.5	0.05		a:0	**0.9992**	**1**	**0.9996**	**0**	1.5
0.5	0.05		a:5	**0.9992**	**1**	**0.9996**	**0**	1.5
0.5	0.05		a:-5	**0.9992**	**1**	**0.9996**	**0**	1.5
0.5	0.1		a:0	**1**	**0.99841**	**0.99921**	**0**	**0**
0.5	0.1		a:5	**1**	**0.99841**	**0.99921**	**0**	**0**
0.5	0.1		a:-5	**1**	**0.99841**	**0.99921**	**0**	**0**
0.5	1.0		a:0	**0.87671**	**1**	**0.93431**	**0.00042**	270
0.5	1.0		a:5	**0.87671**	**1**	**0.93431**	**0.00042**	270
0.5	1.0		a:-5	**0.87671**	**1**	**0.93431**	**0.00042**	270

Tabelle A.28: Ergebnisse der Datenfusion für das Inject Lidar Szenario und der Methode Multi-Level Dempster Shafer.

Kombination				**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.25	0.05		a:0	0.26427	**1**	0.41806	0.01604	5257.5
0.25	0.05		a:5	0.26427	**1**	0.41806	0.01604	5257.5
0.25	0.05		a:-5	0.26427	**1**	0.41806	0.01604	5257.5
0.25	0.1		a:0	0.23327	**1**	0.3783	0.01389	6222
0.25	0.1		a:5	0.23327	**1**	0.3783	0.01389	6222
0.25	0.1		a:-5	0.23327	**1**	0.3783	0.01389	6222
0.25	1.0		a:0	0.17112	**1**	0.29224	0.0146	9300
0.25	1.0		a:5	0.17112	**1**	0.29224	0.0146	9300
0.25	1.0		a:-5	0.17112	**1**	0.29224	0.0146	9300
0.5	0.05		a:0	0.23388	**1**	0.3791	0.01889	6186
0.5	0.05		a:5	0.23388	**1**	0.3791	0.01889	6186
0.5	0.05		a:-5	0.23388	**1**	0.3791	0.01889	6186
0.5	0.1		a:0	0.21925	**1**	0.35965	0.01503	6741
0.5	0.1		a:5	0.21925	**1**	0.35965	0.01503	6741
0.5	0.1		a:-5	0.02092	0.07607	0.03281	0.01503	6741
0.5	1.0		a:0	0.16162	**1**	0.27826	0.01553	9960
0.5	1.0		a:5	0.16162	**1**	0.27826	0.01553	9960
0.5	1.0		a:-5	0.16162	**1**	0.27826	0.01553	9960
1.0	0.05		a:0	0.21611	**1**	0.35541	0.021	6877.5
1.0	0.05		a:5	0.21611	**1**	0.35541	0.021	6877.5
1.0	0.05		a:-5	0.21611	**1**	0.35541	0.021	6877.5
1.0	0.1		a:0	0.20839	**1**	0.3449	0.01602	7191
1.0	0.1		a:5	0.20839	**1**	0.3449	0.01602	7191
1.0	0.1		a:-5	0.20839	**1**	0.3449	0.01602	7191
1.0	1.0		a:0	0.16203	**1**	0.27887	0.01555	9930
1.0	1.0		a:5	0.16203	**1**	0.27887	0.01555	9930
1.0	1.0		a:-5	0.16203	**1**	0.27887	0.01555	9930

Tabelle A.29: Ergebnisse der Datenfusion für das Delay Manual Steer Szenario und der Methode Multi-Level Dempster Shafer.

Kombination				Ergebnis				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.1	0.05		a:0	0.68624	**0.99605**	**0.81262**	**0.00264**	865.5
0.1	0.05		a:5	0.68624	**0.99605**	**0.81262**	**0.00264**	865.5
0.1	0.05		a:-5	0.68624	**0.99605**	**0.81262**	**0.00264**	865.5
0.1	0.1		a:0	0.59335	**0.96197**	0.73398	**0.00279**	1248
0.1	0.1		a:5	0.59335	**0.96197**	0.73398	**0.00279**	1248
0.1	0.1		a:-5	0.59335	**0.96197**	0.73398	**0.00279**	1248
0.1	1.0		a:0	0.55682	0.76562	0.64474	**0.00185**	1170
0.1	1.0		a:5	0.55682	0.76562	0.64474	**0.00185**	1170
0.1	1.0		a:-5	0.55682	0.76562	0.64474	**0.00185**	1170
0.25	0.05		a:0	0.6411	**0.99682**	0.78034	**0.00322**	1053
0.25	0.05		a:5	0.6411	**0.99682**	0.78034	**0.00322**	1053
0.25	0.05		a:-5	0.6411	**0.99682**	0.78034	**0.00322**	1053
0.25	0.1		a:0	0.61622	**0.99046**	0.75976	**0.0026**	1164
0.25	0.1		a:5	0.61622	**0.99046**	0.75976	**0.0026**	1164
0.25	0.1		a:-5	0.61622	**0.99046**	0.75976	**0.0026**	1164
0.25	1.0		a:0	0.51402	0.85938	0.64327	**0.00246**	1560
0.25	1.0		a:5	0.51402	0.85938	0.64327	**0.00246**	1560
0.25	1.0		a:-5	0.51402	0.85938	0.64327	**0.00246**	1560
0.5	0.05		a:0	0.47626	**0.9992**	0.64506	0.00632	2068.5
0.5	0.05		a:5	0.47626	**0.9992**	0.64506	0.00632	2068.5
0.5	0.05		a:-5	0.47626	**0.9992**	0.64506	0.00632	2068.5
0.5	0.1		a:0	0.5016	**0.99841**	0.66773	**0.00418**	1872
0.5	0.1		a:5	0.5016	**0.99841**	0.66773	**0.00418**	1872
0.5	0.1		a:-5	0.5016	**0.99841**	0.66773	**0.00418**	1872
0.5	1.0		a:0	0.40559	0.90625	0.56039	**0.00403**	2550
0.5	1.0		a:5	0.40559	0.90625	0.56039	**0.00403**	2550
0.5	1.0		a:-5	0.40559	0.90625	0.56039	**0.00403**	2550

Tabelle A.30: Ergebnisse der Datenfusion für das Reorder Camera Szenario und der Methode Multi-Level Dempster Shafer.

Kombination				**Ergebnis**				
P(A)	**Intvl.**	**Algo.**	**Konfig.**	**Prec.**	**Recall**	**F1**	**FPR**	**FP/h**
0.1	0.05		a:0	0	0	0	0.0138	4521
0.1	0.05		a:5	0	0	0	0.0138	4521
0.1	0.05		a:-5	0	0	0	0.0138	4521
0.1	0.1		a:0	0	0	0	0.01251	5610
0.1	0.1		a:5	0	0	0	0.01251	5610
0.1	0.1		a:-5	0	0	0	0.01251	5610
0.1	1.0		a:0	0	0	0	0.00937	5940
0.1	1.0		a:5	0	0	0	0.00937	5940
0.1	1.0		a:-5	0	0	0	0.00937	5940
0.25	0.05		a:0	0	0	0	0.01735	5685
0.25	0.05		a:5	0	0	0	0.01735	5685
0.25	0.05		a:-5	0	0	0	0.01735	5685
0.25	0.1		a:0	0	0	0	0.01264	5661
0.25	0.1		a:5	0	0	0	0.01264	5661
0.25	0.1		a:-5	0	0	0	0.01264	5661
0.25	1.0		a:0	0	0	0	0.00954	6090
0.25	1.0		a:5	0	0	0	0.00954	6090
0.25	1.0		a:-5	0	0	0	0.00954	6090
0.5	0.05		a:0	0	0	0	0.01896	6214.5
0.5	0.05		a:5	0	0	0	0.01896	6214.5
0.5	0.05		a:-5	0	0	0	0.01854	6075
0.5	0.1		a:0	0	0	0	0.01401	6282
0.5	0.1		a:5	0	0	0	0.01401	6282
0.5	0.1		a:-5	0	0	0	0.01386	6213
0.5	1.0		a:0	0	0	0	0.01071	6780
0.5	1.0		a:5	0	0	0	0.01071	6780
0.5	1.0		a:-5	0	0	0	0.01071	6780

Glossar

Bandbreite Die Bandbreite bezeichnet die Menge an übertragenen Daten.

CAN Das Controller Area Network Kommunikationsprotokoll wird für die Echtzeitfähige Kommunikation vor allem zwischen ECUs eingesetzt.

Confusion-Matrix Ein Confusion-Matrix (zu deutsch: Verwechslungs-Matrix) bezeichnet eine Tabelle, in der angegeben wird, welche Verwechslungen es bei einer Klassifizierung gab.

Dictionary Ein Dictionary bezeichnet eine Datenstruktur in der Programmiersprache Python. Es besteht aus einer Menge von Schlüsseln, die auf eine Menge von Werten abbildet.

ECU Electronical Control Units sind digitale Komponenten, die in Fahrzeugnetzwerken eingesetzt werden, um Funktionen wie die Beleuchtung oder das Entertainment zu kontrollieren.

Frame Ein Frame bezeichnet eine Übertragungseinheit auf der Sicherungsschicht (Layer 2) im ISO/OSI-Referenzmodell. Oft Synonym mit dem Begriff Paket genutzt.

Jitter Der Jitter ist ein Maß dafür die zeitliche Schwankung bei der Übertragung von Signalen, bspw. von Paketen.

JSON Die JavaScript Object Notation ist ein Dateiformat, das zur Speicherung und Übertragung von Daten eingesetzt werden kann.

Link-Layer Der Link-Layer (Sicherungsschicht) bezeichnet die zweite Schicht im ISO/OSI-Referenzmodell .

Map Eine Map bezeichnet eine Datenstruktur, die aus einer Menge von eindeutigen Schlüssel-Wert-Paaren besteht.

Median-Filter Der Median Filter bezeichnet in der Daten- oder Signalverarbeitung einen Filter, der das Signal glätten soll ohne die Position von Sprüngen im Signal zu verändern.

Netzwerkmetrik Eine Netzwerkmetrik ist ein Wert, der die Kommunikation in einem Netzwerk quantifizieren kann. In dieser Arbeit gibt es die vier Netzwerkmetriken: Bandbreite, Jitter, die durchschnittliche Größe eines Frames und den durchschnittlichen Abstand zwischen Frames.

Page-Fault Ein Page-Fault tritt auf, wenn ein Prozess auf temporär in den Swap-Space ausgelagerten Speicher zugreifen möchte. Da in diesem Fall auf die Festplatte zugegriffen werden muss, können viele Page-Faults zu einer ineffizienten Speichernutzung führen.

Paket Ein Paket bezeichnet eine Übertragungseinheit auf der Vermittlungsschicht (Layer 3) im ISO/OSI-Referenzmodell. Oft Synonym mit dem Begriff Frame genutzt.

Parsing Das parsing bezeichnet im Kontext von Dateien das lesen dieser, also die Umwandlung des Dateitexts in ein geeigneteres, anwendungsspezifisches Format, bspw. die Umwandlung einer JSON - Datei in ein Python-Dictionary.

Prozess-Pool Prozess-Pools sind ein beliebtes Mittel zur Parallelisierung. Sie bestehen aus einer beliebigen Anzahl von Worker-Prozessen, denen Tasks zugewiesen und ausgeführt werden können.

STRIDE Das Akronym STRIDE ist ein von Microsoft vorgestelltes Modell für Sicherheitsrisiken und steht für Spoofing, Tampering, Repudiation, Information disclosure, Denial of service und Elevation of privilege.

Swap-Space Der Swap-Space bezeichnet den Bereich, den das Betriebssystem für die virtuelle Speicherverwaltung nutzen kann. Hierhin können ungenutzte Teile des Speichers ausgelagert werden, um den Speicher effizienter nutzen zu können.

Switch Ein Switch bezeichnet ein aktives Netzwerkgerät, das Netzwerksegmente miteinander verbindet.

UDP Das User Datagram Protocol ist ein Netzwerkprotokoll, auf der Vermittlungsschicht des ISO/OSI-Referenzmodells und wird für die Übertragung von Paketen ohne n verwendet.

XML Die Extensible Markup Language wird zur Speicherung, Strukturierung und Übertragung von Daten verwendet und wird als Dateiformat häufig für Konfigurationsdateien eingesetzt.

Zonal-Controller Der Zonal-Controllerist eine dezentrale Steuerungskomponente in einem Fahrzeug. Sie ist dafür verantwortlich, die Komponenten in einer bestimmten Zone im Fahrzeug zu kontrollieren .

Erklärung zur selbständigen Bearbeitung

Hiermit versichere ich, dass ich die vorliegende Arbeit ohne fremde Hilfe selbständig verfasst und nur die angegebenen Hilfsmittel benutzt habe. Wörtlich oder dem Sinn nach aus anderen Werken entnommene Stellen sind unter Angabe der Quellen kenntlich gemacht.

Ort	Datum	Unterschrift im Original