Federated Intrusion Detection In Autonomous Transportation Systems

Seraphin Zunzer

A thesis submitted to the
Faculty of Electrical Engineering and Computer Science
of the
Technical University of Berlin
in partial fulfillment of the requirements for the degree
Bachelor of Computer Science

Berlin, Germany
March 22, 2023

Main supervisor:

Prof. Dr. Dr. h.c. Sahin Albayrak, Technical University of Berlin

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und eigenhändig sowie ohne unerlaubte fremde Hilfe und ausschließlich unter Verwendung der aufgeführten Quellen und Hilfsmittel angefertigt habe.

Berlin, den

Zusammenfassung

Die zunehmende Automatisierung des Transportsektors und anderer sicherheitskritischer Infrastrukturen geht mit einem großen Problem einher: Aufgrund einer stetig steigenden Anzahl von Cyberangriffen weltweit und damit einhergehende finanzielle Verluste, Imageschäden und Ausfälle kritischer Infrastruktur sind Unternehmen sowie Regierungen dazu gezwungen, Erkennungs- und Gegenmaßnahmen zu entwickeln und umzusetzen. Häufig ist das Ziel, solche Systeme hochgradig zu automatisieren, da die Menge an sicherheitsrelevanten Ereignissen die Kapazität von menschlichen Administratoren übersteigt. Intrusion Detection Systeme erkennen automatisch sicherheitsrelevante Ereignisse, generieren entsprechende Alarmmeldungen und sind heutzutage ein wichtiger Bestandteil der Cybersicherheit komplexer IT-Systeme. Moderne Intrusion Detection Systeme haben allerdings nur eine begrenzte Wirksamkeit gegen unbekannte Angriffe und eine unzureichende Anpassungsfähigkeit an unterschiedlich komplexe Umgebungen. Außerdem mangelt es häufig an Daten zum Trainieren komplexer Intrusion Detection Systeme, da diese meistens auf verschiedenste Geräte verteilt anfallen und nicht ohne weiteres zusammengeführt werden können.

Intrusion Detection Systeme könnten mit einem der neuesten Konzepte des maschinellen Lernens, dem sogenannten Federated Learning, kombiniert werden, um die genannten Probleme zu lösen und um eine wesentlich verbesserte automatische Erkennung sicherheitsrelevanter Ereignisse zu erzielen. Durch den Austausch von Modelldaten stärken sich im Federated Learning die Modelle verschiedener Geräte untereinander und erreichen damit eine höhere Genauigkeit in der Analyse von sicherheitsrelevanten Ereignissen. In der vorliegenden Arbeit werden verschiedene Ansätze von Federated Learning für die Nutzung der Angriffserkennung untersucht. Es wird ein Federated Learning Framework implementiert, das die zuverlässige Erkennung von Cyberattacken in Netzwerken erlaubt, trotzdem vollständig flexibel auf unterschiedliche Anwendungsgebiete zugeschnitten werden kann und damit viel Potenzial für zukünftige Weiterentwicklungen bietet. Anschließend wird für die Evaluation des Frameworks ein Autoencoder verwendet, um im Kontext eines autonomen Transportsystems den Federated Learning Ansatz zu evaluieren. In umfangreichen Experimenten mit dem CIC-IDS17-Datensatz und mit gesammelten realistischen Netzwerkdaten erreicht unser Algorithmus eine hohe Genauigkeit von 90% und eine Erkennungszeit von durchschnittlich weniger als 0,5 Sekunden für Angriffe. Außerdem haben wir gezeigt, dass Federated Learning eine Verbesserung der True-Positive-Rate von bis zu 20% für einzelne Geräte gegenüber einer klassischen Trainingsumgebung ermöglicht.

Abstract

The increasing automation of the transportation sector and other safety-critical infrastructures is accompanied by a major problem: Due to a steadily increasing number of cyberattacks all over the world and the caused financial losses, image damages and failures of critical infrastructure, companies as well as governments are forced to develop adequate countermeasures. Intrusion Detection Systems are one of the most essential parts of cybersecurity and a highly effective mechanism to fight the growing threat. Often it is necessary to highly automate such systems, as the amount of security related events exceeds the capacity of human administrators. Intrusion detection systems automatically detect security-related events, generate appropriate alerts and are nowadays an important part of cybersecurity of complex IT systems. However, modern intrusion detection systems have a limited effectiveness against unknown attacks and insufficient adaptability to environments of varying complexity. In addition, the neccessary training data is often heterogeneous, spread across multiple locations, and cannot be shared due to regulatory, competitiveness, or privacy reasons.

Intrusion detection systems could be combined with one of the latest machine learning concepts, called federated learning, to solve the aforementioned problems. Federated learning enables multiple parties to construct a machine-learning model collaboratively so that the models of different devices strengthen each other and thus achieve higher accuracy in the analysis of safety-relevant events. In this thesis, we propose a framework using the popular federated averaging algorithm for the aggregation of autoencoder weights to detect intrusions in modern autonomous transportation systems. We present the customizable machine learning model, the federated optimization algorithm, and our own communication protocol with encrypted data transmission. In extensive experiments on the CIC-IDS17 dataset and our own collected real-world network data, the algorithm achieves a high accuracy of around 90% in average, a detection time of less than 0.5 seconds for anomalies, and a data throughput of around 150 samples per second. We showed that federated learning allows advancements of up to 20% in true positive rate for individual participating clients over a non-federated training setting.

Acknowledgements

First of all, I am grateful to my advisor Fabian Hofmann, who offered me to work on this enlightening topic. His support, suggestions, and constructive comments over the last months were of great help. I would also like to thank Prof. Dr. Dr. h.c. Sahin Albayrak for his support of my thesis. Finally, I must express my very profound gratitude to the whole BeIntelli-Team for their passionate assistance by providing their infrastructure and knowledge. This accomplishment would not have been possible without their future-oriented teamwork.

Contents

List of Figures

List of Tables

List of Abbreviations

1
Introduction

In the modern internet, countless computers are connected together, playing an essential role in the life of almost every person. Cars become smarter and start to communicate with each other, while autonomous transportation systems have the opportunity to show their potential to reshape everyday mobility. The increasing number of automation in the transport sector is accompanied by a big problem: A rising number of cyberattacks all over the world and the highest possibility of being exposed to tremendously harmful and cost-intensive attacks [21, 47]. Half of all published cyberattacks targeted against cars occurred in 2021 alone [77]. Therefore, researchers try to find new ways to detect and prevent intrusions to ensure the security of software, devices, and humans. Due to the increasing amounts of data transmitted in modern networks, it is no longer possible for humans to monitor these data streams for malicious activity manually. Therefore, an improved automation is unavoidable to detect and prevent threats before they can cause serious damage. An intrusion detection system (IDS) is one of the most essential parts of modern cybersecurity and a highly effective mechanism to fight the growing threat [109]. Using different types of algorithms, from simple pattern recognition to advanced machine learning, IDS became increasingly precise in the last decades. Some developments can even detect worldwide unknown zero-day attacks [53]. However, IDS have several problems in real applications, where data is heterogeneous and spread across multiple locations, and cannot be shared due to regulatory, competitiveness, or privacy reasons [94]. In situations where sharing would be possible, the high amount of data makes sharing costly and time-consuming.

With one of the latest concepts in machine learning, the so-called federated learning (FL), we could solve the mentioned problems of traditional IDS: FL enables multiple parties to collaboratively construct a machine-learning model while keeping their training data private [110]. Therefore, multiple trained models from different heterogeneous devices (in our case: vehicles in a global transportation system) are aggregated into one intelligent model, including everything that has been learned before on separate clients. Vehicles using FL could update their IDS autonomously as soon as one device detects new malicious behavior. By bringing model training to the vehicles as well, we can contrive highly secure mobility in a modern and reliable transportation system for the future.

In this thesis, we propose an algorithm using FL for intrusion detection in modern, distributed, and autonomous transportation systems. Since the needed data comes from a safety-critical area, there is nearly no open-source data from other companies or publications. Therefore, we decided to evaluate the performance of our algorithm using the Canadian Institute for Cybersecurity Intrusion Detection Evaluation 2017 (CIC-IDS17) dataset and real network data mixed with simulated attacks. The real-world dataset was collected from the roadside infrastructure of a real intelligent transportation system. As a demonstration, we implemented an autoencoder for the detection of the anomalies and aggregated multiple trained client models using the popular federated averaging (FedAVG) algorithm. We will estimate the performance by comparing the achieved results and the system's advantages to conventional intrusion detection approaches, so we can weigh up whether the performance in recognition of intrusions improves due to FL.

The rest of this work is organized as follows: In chapter 2 we start with the background information needed for understanding the approaches and the challenges that come with autonomous transportation systems. Then we will describe the most common attack types, recent cyberattacks on transportation systems, and the foundations of IDS and FL. In chapter 3 we will continue with the state of the art, where we discuss recent developments in intrusion detection in connection with FL and the latest challenges. Proceeding with our own idea and the presentation of the implementation of a federated intrusion detection system (F-IDS) in chapter 4, we will take a closer look at the realization inside an autonomous transportation system. We will conduct an extensive evaluation on the CIC-IDS17 and the real-world dataset in chapter 5 and a critical discussion of our results. Finally, chapter 6 concludes this thesis, with potential future work being highlighted.

2
Background

In this chapter, we will give a short introduction to autonomous transportation systems, explain the basics of intrusion detection and federated learning (FL), and give examples of recent developments in autonomous transportation systems.

2.1 Autonomous Transportation Systems

From day to day, the topic of autonomous driving is becoming more important. In the future, autonomous vehicles and driverless transportation systems for people and goods will probably dominate the townscape [72]. According to experts, autonomous vehicles will travel 66% of the total passenger kilometers in 2040 [72]. These vehicles are driving on fixed or even flexible routes in industrial areas, in cities, and even in the surrounding rural areas with a large proportion of these being not owned by private individuals. In literature, there are different application possibilities suggested, where autonomous transportation systems can be established [36]. Vuchic et al. propose the following classification of everyday transportation systems, divided into three categories [105]:

Private Transportation: Includes privately owned vehicles (e.g. cars or bicycles) for personal use on public streets. Each vehicle serves a person or a related group.

For-hire Transportation: Operators provide transportation systems and are available to parties that hire them, e.g., taxi and car sharing.

Public Transportation: Contains vehicles and trains that are available for the use of anyone who pays the established fee. These vehicles normally operate on fixed routes and with fixed schedules (e.g., bus and rail transit).

Autonomous driving could emerge in all of the proposed areas. Since the willingness of the population to use autonomous trains and cars is higher than in other transportation methods like airplanes and ships [49], we want to focus on these modes of transport in this thesis. In this way, we are addressing a large part of daily mobility and the current research priorities. Researchers are incrementally adding active safety and self-driving features to traditional vehicles and trains

by automating acceleration, braking, and steering. The various levels that are reached by the incremental addition of these features were formally defined by the Society of Automotive Engineers (SAE), which developed an industry-standard scale from zero to five to describe the different levels of automated driving (J3016, [85]):

Level 0 - No Automation: The driver is completely responsible for controlling the vehicle.

Level 1 - Driver assistance: Automated systems start to take control of the vehicle in certain situations to assist the driver.

Level 2 - Partial Automation: The vehicle has a greater awareness of its surroundings, allowing it to perform more complex functions that pair steering with acceleration and braking.

Level 3 - Conditional Automation: Autonomous driving is limited to certain vehicle speeds, road types, and weather conditions.

Level 4 - High Automation: The vehicle's autonomous driving system can monitor the driving environment and handle all driving functions on routine routes.

Level 5 - Full Automation: No driver is required behind the steering wheel.

The introduction of autonomous vehicles is expected to generate many social and economic benefits, such as improved traffic safety due to driver assistance, higher service quality, and improved efficiency [14]. At the same time, there is an improved mobility offer for both young and old people and also improved accessibility in rural areas [95]. Additionally, autonomous transportation systems could enhance productivity in the transportation and logistics industries while reducing congestion and fuel consumption. All of these factors indicate that autonomous vehicles will be of great importance in the future. In recent studies, most users give positive feedback for autonomous transportation systems [73, 49]. They were satisfied with the additional departures of autonomous buses and felt safe while traveling. However, this requires a precise and reliable communication with other vehicles and a high awareness of the car's surroundings in the background. Therefore, new developments in self-driving vehicles collect information from the environment using various roadside sensors that the intelligent infrastructure of the future offers. The data is processed using various approaches, including those using machine learning [95]. This allows the vehicle to use the data obtained from the sensors, even though they might be inaccurate or interrupted. After analyzing the current state and making a decision, connectivity allows autonomous vehicles to communicate with each other (vehicle-to-vehicle (V2V)), the road infrastructure (vehicle-to-infrastructure (V2I)) or everything in the surrounding area (vehicle-to-everything (V2X)). There are many technologies to enable different types of communication, e.g., Bluetooth is useful for short-range communication, while Wi-Fi could be used as medium-range and LTE or 5G Networks as long-range communication technologies [75]. In the following, we present a few crucial projects aimed at the development of the mobility of the future.

2.1.1 BeIntelli

The Beintelli[1] Project, funded by the Federal Ministry for Digital Affairs and Transport of Germany, is building an urban test site in Berlin, developing an infrastructure of different autonomous transportation systems in the surrounding area of the Technical University of Berlin. Using state-of-the-art sensor technology like 3D-Lidar, radar, and camera systems, self-driving vehicles will navigate through the test area. BeIntelli uses different autonomous vehicles like cars and buses and plans to extend the fleet in the future, e.g., with drones and delivery robots. Furthermore, bus stops for public transport are used to inform people about the BeIntelli Project and teach bystanders useful facts concerning artificial intelligence. The goal is the creation of the infrastructure for completely autonomous operating vehicles. In section 4.8 we will explain the individual components of the BeIntelli system in detail and highlight the application sites of a federated intrusion detection system (F-IDS) in such a network.

2.1.2 Autonomous Railway Systems

Similarly, the railway industry drives automated train driving forward. Alstom, one of the biggest international train manufacturers is developing driverless and driver assistance technologies for trains, ranging from trams up to heavy locomotives, to improve safety, as well as service quality and efficiency [45]. According to themselves, they are one of the leading companies in the field of autonomous train systems by developing incident prevention systems, eco-driving support to reduce energy consumption, and solutions for improved passenger comfort by autonomous speed control to increase punctuality. In April 2017, Alstom achieved a remarkable breakthrough with the installation of an automatic train operating system on Europe's busiest subway line carrying 1.2 million passengers per day in Paris [10]. The implemented technology controls the speed of the train and is an addition to existing train spacing systems, making the operation fully automatic. Alstom plans to increase speed and punctuality on this line in the future by optimizing its systems. Another important research field is the automation of trams. There are significantly more challenges for artificial intelligence and autonomous driving in the open environment of a tramway than inside subways [92]. Therefore, the German train producer Siemens Mobility GmbH is currently testing autonomous trams in Potsdam, Germany. The first stage was finished in October 2019, realizing a completely automated tram depot [92].

2.2 Threats of Cyberattacks

In surveys on autonomous driving, people are mostly concerned about the hacking and misuse of software of the vehicles [58]. As described in the previous section, autonomous vehicles use many kinds of sensors to gather information about the environment. These sensors communicate with the internal on-board units (OBUs) and other devices in and around the vehicle. If the communication between infrastructure, sensors, and the OBU is maliciously controlled or interrupted, the vehicle could cause serious accidents [40]. Some of the most common attack types on a self-driving vehicle that may allow attackers to manipulate the vehicle or suppress data packets transmitted in the network are for example:

[1] https://be-intelli.com/

Distributed denial of service (DDoS): Multiple systems attempt to overload a victim's bandwidth or resources. Often the result of multiple distributed systems (e.g., a botnet) that try to flood the target system by creating an enormous amount of network traffic [43].

Ransomware: A program restricts access to the user files by encrypting them and the attacker demands a ransom for the decryption [51].

Botnet: A high number of internet-connected devices are used to perform various attacks automatically. It can be used to steal data, send spam and give the attacker access to the device and its connection [90].

Brute Force: Brute force attacks are frequent in networks as they try to break into accounts with weak usernames and passwords by trying common password and username combinations [90].

Infiltration: A network is usually infiltrated from the inside by exploiting vulnerable software. After successful infiltration, various attacks are launched on the victim's network [90].

GPS Spoofing: A GPS simulator is used to generate disturbing radio signals to convince the GPS receiver inside the vehicle that it is in another location [71].

Tracking: Attackers try to collect personal information, e.g., the position of the vehicle [71].

Illusion attack: Attacker creates a specific traffic situation and e.g., sends false traffic warning messages to control or manipulate other vehicle's behavior [71].

In the last few years, many cyberattacks on autonomous vehicles have been reported. The damage of such attacks can be enormous: In 2022, an automotive supplier had its systems breached by three different ransomware attacks in two weeks [39]. The attackers used the internet to gain access to the victim's private network for a few days in April and May. Then, the ransomware was distributed across the private network and by deleting shadow copies and clearing out the event logs on the compromised systems, recovery attempts became nearly impossible. In the end, the attackers completely encrypted more than a dozen systems and pressurize the victim to pay a ransom for release. Additionally, in 2020, the Japanese car manufacturer Honda became a victim of a cyber-attack. According to the company, a virus had spread throughout its network, causing the temporary shutdown of production facilities, as well as the unavailability of the customer and financial services [101]. In Texas, a hacker remotely attacked more than 100 vehicles [80], which were no longer drivable or started honking unstoppably. Even the surrounding infrastructure is not secure enough, as hackers have demonstrated that traffic lights are very easy to manipulate [34]. At the same time, also governments and cities are affected by the increasing number of cyberattacks. After a ransomware attack on a district administration in 2021, a "major cyber emergency" was declared in Germany for the first time. Public services were no longer available even months after the attack and nearly no normal operation was possible [21]. Other cities are also repeatedly affected by cyberattacks [21, 88], whereby attacks such as DDoS and ransomware not only have a devastating impact on such entities but also on businesses, disrupting the delivery of critical services [21]. To protect our systems against future attacks, it is necessary to not only develop countermeasures and enhanced cybersecurity but also methods to detect these attacks in the first place and generate respective alerts for them. A really effective method is an intrusion detection system (IDS), which we will describe in detail in the following section.

2.3 Intrusion Detection Systems

From simple pattern matching over classical machine learning to complex neural networks, the main task of an intrusion detection system (IDS) is to monitor activities and report malicious behavior. By monitoring traffic in a network or received on devices, IDS turned out to be one of the most powerful methods that can handle network intrusions [7], by being able to detect and stop unknown and suspicious activities that cannot be automatically detected by firewalls or similar technologies. In the following, we want to explain different types of IDS in detail.

Host- & Network-based IDS Based on the location of the IDS, researchers differentiate between two types of intrusion detection. Host-based intrusion detection could be used when data of an individual calculation unit should be analyzed in log files or system activity data, while a network-based IDS analyses the traffic transferred between several clients in the network to detect attacks launched by outside hackers that want to gain unauthorized access [23]. For this thesis, we will focus on network data exchanged between the host and other (potentially harmful) devices.

Anomaly- & Misuse-based detection Furthermore, IDS can be classified into two categories. In (signature-based) misuse detection systems, the signature patterns of known attacks are stored in a database. By matching them with the monitoring metrics, the intrusion can be detected. Consequently, this method cannot identify zero-day attacks [113] and requires a high level of domain expertise to maintain the database with up-to-date signatures. Anomaly-based systems analyze any activity in the network for unexpected or unusual patterns using machine learning classifiers and neural networks. Anomaly-based systems are much more reliable when it comes to new, sophisticated attacks, but suffer from a higher false alarm rate [16], caused by the fast-changing network environment. The patterns of normal traffic can slightly differ or change over time which the anomaly-based system has to adapt to [112].

Supervised, Semi-supervised & Unsupervised Learning To train and evaluate an IDS, researchers use different datasets (see section 4.6). After training an IDS with datasets like these, the system should detect intrusions in the network. If an IDS is trained with a dataset containing both, normal and anomalous labels, it is called a supervised approach. When assumptions are made about the training data, such as only normal traffic is used for training, this is called a semi-supervised approach [23]. If unlabeled training data is used, it is called an unsupervised approach [16]. In supervised learning, the algorithm learns by making predictions on the training dataset repeatedly and adjusting the model weights for the right answer, while in unsupervised learning, the algorithm tries to find the underlying structure of unlabeled data independently [31]. Supervised techniques can benefit from a lower false positive rate because the researchers can exactly define what is normal and abnormal. However, supervised learning models can have a time-consuming training [31], and are typically used in traditional intrusion detection approaches. In most real-world applications, security experts do not have any labeled datasets, and labeling the data would be unimaginably time-consuming and would require lots of domain expertise. The use of existing datasets is also not possible since static datasets are often insufficient and impracticable in a fast-changing real-world environment that is much more challenging than the ones depicted by the original dataset [91, 13].

Types of Anomalies Anomalies can be divided into three groups: Point, Contextual, and Collective anomalies. It is important to correctly identify the type of anomaly we want to detect in order to subsequently select the most appropriate machine learning method for its detection. Chandola et al. described the types of anomalies in [23]:

Point Anomalies: A point anomaly is characterized as a single value that is abnormal in comparison to the rest of the data. In our experimental setup, e.g., Data Leakages are point anomalies due to their particularly large packet size, caused by an attacker stealing as much information as possible (see section 4.8). A good detection by IDS is usually possible.

Contextual Anomalies: A contextual anomaly is characterized as a data instance that is anomalous in a specific context (but not otherwise). In our real-world dataset, the installation of an attack tool can be considered as a contextual anomaly, since the installation of the tool in connection with further exploitation of vulnerabilities of the system can be seen as anomalous, however, the installation of tools by itself is not directly malicious (see section 4.8).

Collective Anomalies: A collective anomaly is characterized as a collection of related data that is anomalous regarding the rest of the dataset. However, a single data instance could be normal by itself, while the occurrence together of such instances is an anomaly [23]. DDoS is such an anomaly type and is difficult to detect for IDS.

Online & Offline Processing Nowadays, there are two different ways in which data can be processed: Offline and online. The processing could be the training of machine learning models or the prediction of labels. In offline processing, the data is collected over a period of time and stored in static datasets or batches. Then, the machine learning model is only trained on this data where the entire dataset has to be available for training at the beginning [4, 54]. In online processing, only the latest collected data samples are available for the training and prediction [4]. For example, online learning trains a model on data instances that arrive in a continuous data stream either in small batches (a subset of points) or one after the other (real-time) [26, 41]. A frequent challenge in most situations is the detection of anomalies in real time as new data values arrive in data streams. In the case of transportation systems, online processing is necessary due to different reasons. First, there is not enough storage for all the received data. Furthermore, in real-world time series, it could happen that the trends in the underlying data stream change over time. A fixed previously trained model would not be able to notice these changes. We also want to prevent harmful packages from reaching areas where they could have a negative impact, therefore we need to immediately detect new intrusions in autonomous transportation systems.

2.4 Federated Learning

Now that we have clarified the basics of intrusion detection, we will have a detailed look at federated learning (FL). For the training of modern IDS, datasets are required. Unfortunately, most companies and organizations do not share novel datasets for several reasons: First, making internal confidential information available to competitors could lead to loss of market placement. Second, attackers could use this information for future attacks, getting insights into the intrusion detection algorithms of a company (this is also known as restricted knowledge white box adversarial attacks, where incoming data is intentionally manipulated by an adversary to cause incorrect model predictions [6]). Third, privacy protection rules released by governments (e.g., the General Data Protection Regulation enacted by the European Union [27]) aim to protect individuals regarding the processing of personal data by restricting the free movement of the data. Therefore, privacy-related data can only be shared under certain circumstances, but features have to be elaborately anonymized, and it must be ensured that individuals cannot be identified by correlation. Additionally, analysts assume that in 2025, more than 75% of enterprise-generated data is created and processed outside a traditional centralized data center or the cloud [70]. Using edge computing, services even including artificial intelligence can be provided close to the source of data [22]. The data is stored and directly processed at the edge of the network which significantly increases speed, real-time processing, and security. In addition, it can solve the problem of excessive energy consumption in cloud computing, has a lower operational cost, and can significantly reduce the overall bandwidth [22].

To solve the bottleneck in data availability and problems coming with distributed datasets, federated learning (FL) was invented by Google in 2015 [56]. The goal was to learn about multiple smartphone users' input behavior to improve the accuracy of the next word prediction of the keyboard without violating privacy rights [56]. FL algorithms build machine-learning models based on different heterogeneous datasets that are distributed across multiple devices (see Figure 2.1) [56]. In general, multiple local models are trained on clients and sent to an aggregation server under consideration of data privacy, i.e., by encryption or using other techniques to ensure that third parties cannot retrieve any private information (see chapter 3). Clients could be placed in different locations, have different calculation power installed, could be unreliable, and may have a slow internet connection [55]. On the server, a global model is created that compresses all the trained knowledge. The most popular FL algorithm called federated averaging (FedAVG) simply averages the local model weights [67]. There are many other ways for the aggregation of the model itself, the weights, or the trained gradients, e.g., by dividing the model into private personalized layers and base layers that are transmitted and aggregated by the server (FedPer)[12] or by matching and merging weights with similar weights, while the layers are trained independently of each other and transmitted to the server (FedMa) [106]. Yang et al. tried to find a general formal description for FL [110]. They assume there are N data owners $C_1, \cdots, C_N$, which want to train a machine learning model together with their own datasets $D_1, \cdots, D_N$. In conventional use cases, all the data is equalized and collected on a central server as $D_{CONV} = D_1 \cup \cdots \cup D_N$ to train the machine learning model M_{CONV}. However, in a FL system, each data owner C_i trains a model M_i without sharing its data D_i with the others or a global instance. All the models $M_1, \cdots, M_N$ are then sent to a global server and aggregated to a global model M_{FED}. The accuracy of M_{FED}, should be very close to the performance of M_{CONV} [110].

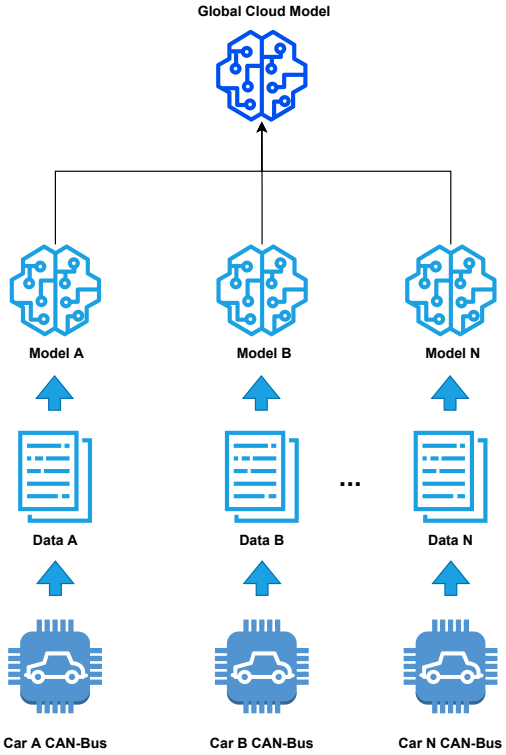


Figure 2.1: The FL process, where multiple clients train a global machine learning model without exchanging individual data [25]

2.4.1 Types of Federated Learning

Lately, Yang et al. proposed three different types of FL methods, based on the distribution of the data features and data samples among the clients [110]. Accordingly, FL can be generally classified as Horizontal FL, Vertical FL, and Federated Transfer Learning.

In Horizontal FL, the datasets owned by each client share similar features but concern different spaces in samples. Yang et al. provided a good example: Assume two regional banks that may have very different user groups based on their regions, and the intersection set of their users is very small. However, both businesses are very similar, so the feature spaces are the same. In real-world scenarios, most datasets stored at different clients contain diverse feature sets but the same sample space. In this case, Vertical FL is applicable, when clients have large overlaps in the sample space but differ in the feature space, i.e., different participants hold different attributes of the same records. According to the example given by Yang et al.: A bank and an e-commerce company are likely to have the same user sets, however, a bank and an e-commerce business record completely different user data. Vertical FL is used for aggregating these different features to build a model with data from both parties. Federated Transfer Learning deals with scenarios in which clients have little overlap in both the sample space and the feature space. Yang et al. described Federated Transfer Tearning with one bank that is located in Europe and an e-commerce company that is located in the United States States. Due to the geographical restrictions, the user groups of both businesses have a small intersection. On the other hand, due to the different business areas, the feature space from both parties is likely to

not overlap. In this case, federated transfer learning is an important extension of the existing FL approaches. In our system, we need the concept of Horizontal FL. The devices differ from the locations in an autonomous transportation system, but all have more or less similar feature spaces since the network traffic always contains similar features.

2.4.2 Synchronous & Asynchronous Federated Learning

Researchers divide the methods of how a FL system updates the global model into two categories: Synchronous or asynchronous [108]. In asynchronous FL, several clients are performing the local client's training simultaneously, while in synchronous learning only one client trains its local model after the other. Recently, Chen et al. proposed an asynchronous online FL framework (ASO-Fed), where several clients perform asynchronous online learning with continuous processing of local data streams [26]. Other research approaches combine both methods: Stripelis et al. introduced a novel energy-efficient semi-synchronous FL protocol [94], that achieves a minimum idle time and a fast convergence by mixing local models periodically. In chapter 4, we describe the functionalities of asynchronous and synchronous FL in detail and explain the main differences concerning the well-known FedAVG optimization algorithm. Based on the findings, we choose the most appropriate method for a federated intrusion detection in autonomous transportation systems.

2.4.3 Federated Learning with Non-IID Data

A major advantage of FL algorithms is that they can also handle unbalanced, non-independent, and identically distributed data (so-called non-IID data). In most real-world applications, data generated from different devices in the network can vary strongly [68]. McMahan et al. have demonstrated that their proposed FedAVG algorithm, which we will later use for our contribution, can work with certain non-IID data [67]. Therefore, in our research datasets with more or less heterogeneous intrusion detection tasks on network traffic, this approach will be sufficient. However, on highly biased data, some federated algorithms suffer from a rapidly decreasing accuracy [114]. Zhao et al. developed a strategy to improve the training of FL settings on non-IID data by creating globally shared subsets [114]. The sharing is performed before the training to reduce communication costs, and to ensure privacy by not sharing the private data of individual clients. Sattler et al. proposed a communication-efficient federated optimization algorithm to improve the performance of FL on non-IID data, that outperforms FedAVG in some aspects [87]. The authors compared their system and FedAVG on a wide variety of different datasets and model architectures, and conclude that the implemented high-frequent low-volume communication is well suited for FL environments that are characterized by low latency and low bandwidth channels between clients and server. Since low latency is not ensured in broad autonomous transportation systems, and also the bandwidth can vary from 5G in the city to no reception in rural regions, we decided that federated averaging offers the most advantages for our implementation.

3

State of the Art

In this chapter, we will present the latest paper and research publications about intrusion detection system (IDS) in transportation systems. Furthermore, we will review the developments of new federated learning (FL) approaches in intrusion detection in other application areas.

3.1 Intrusion Detection in Transportation Systems

There are various application areas for intrusion detection system (IDS), as e.g., in Local Area Networks [7] or on World Wide Web Servers [104]. In the following, we want to focus on the usage of IDS in transportation systems and present the latest research topics.

Vehicular Systems In the last years, automobiles became progressively more connected, increasing the risk of being affected by cyberattacks [77, 80, 52] and making new automated detection methods for cyberattacks necessary. On one side, connected cars that communicate with the infrastructure (vehicle-to-infrastructure (V2I)) and the surrounding environment (vehicle-to-everything (V2X)) improve passenger convenience, passenger safety, and traffic efficiency, but at the same time they introduce new attack possibilities. Since multiple cyberattacks were reported over the last years and more and more companies see their existence threatened by cyberattacks [11], researchers and car manufacturers have started to implement intrusion detection and prevention systems in cars, last but not least also to foster the confidence in these systems [102]. By filtering the data streams exchanged between the car and the outside world, as well as within the automotive bus systems, an IDS can already effectively detect common attacks. Inside the car, the controler area network (CAN) ensures that all electronic control units (ECU) can communicate with each other. CAN is a cheap, lightweight, and complex bus communication protocol that defines a standard for effective and reliable transmission inside the car [44]. The connectivity of cars with the outside world is typically achieved by mobile radio connections with cellular networks like 4G and 5G or wifi providing access to the Internet and backend systems [52]. This communication enables a variety of applications for navigation, the onboard entertainment systems, and the V2I or the V2X communication for higher levels of automated driving [42]. However, the internal and external networks of autonomous vehicles

are vulnerable to cyberattacks and lack of accurate and reliable security mechanisms [44]. In [74], the authors demonstrate how easily regular operations can be disrupted by manipulating exchange protocols or by launching different attacks on the CAN. They conclude that malicious adversaries could cause significant damage to the system by exploiting flaws in the design of the CAN protocol [74]. A concrete experiment was conducted in [59], where the authors attacked a vehicles CAN without any in-depth knowledge by injection of self-constructed packets. The packets were injected into the network over the car's wireless connectivity without any additional equipment. To prevent such cybersecurity incidents in the autonomous transportation systems of the future, the authors strongly recommend implementing smart defense methods like IDS in vehicles.

An IDS could pre-process all the exchanged data and extract certain features to analyze them for malicious traffic. Alsarhan et al. proposed an IDS for vehicular ad-hoc networks containing a support vector machine (SVM) and basic machine learning approaches to select the optimizer of the classifier [9]. The proposed system achieves high results in the evaluation on an old dataset (with the worst optimizer an accuracy of 89%, with the best optimizer an accuracy of 98%), but has never been evaluated or deployed on cars and real-world data. Furthermore, after training the SVM classifier with the static dataset the system becomes vulnerable to novel attacks since there is no mechanism to improve the model over time.With our contribution, we will also evaluate the developed federated intrusion detection system (F-IDS) with real-world network data, and the system will learn over time to be able to better adapt to changing environments.

Taylor et al. proposed an anomaly detection system to identify abnormal behavior of the cars' internal connections with SVM[97]. The system measures and compares current and historical packet timing over a sliding window between the different car components. Then, the average times are compared to historical averages to return an anomaly signal. The system was evaluated using collected CAN bus data from an experimental car, a 2011 Ford Explorer. The traffic was captured simultaneously from the two CAN buses while the car drove at low speed for about five minutes. According to the authors, the IDS may be sufficient for simple use cases, but the performance was highly dependent on the packet timing. An improved solution must be capable of dealing with more complex situations and a lot of traffic between many car components, but therefore, more data is needed. The system is also only designed for periodic traffic which is not a good solution for non-heterogeneous autonomous transportation systems where transmission times could vary or non-periodic new packages could arrive at any time.

Lee et al. analyzed the response performance of vehicle components to detect anomalies [60]. The IDS sends requests to car components every 0.01 second and receives response messages. For intrusion detection, only the response times are measured and analyzed for deviations. This method uses a large amount of limited resources in the restricted environment of in-vehicle networks in which unnecessary messages are sent back and forth, but at least the proposed system can successfully detect the most dangerous attacks in the used evaluation dataset.

Wang et al. proposed a deep learning based IDS for end-to-end network traffic in smart vehicles [107]. To improve the real-time performance, the authors implemented a threshold-based mechanism to update the model occasionally. The effectiveness of the proposed method was evaluated using a real network traffic dataset, but unfortunately, this approach was highly computationally intensive. Therefore, the authors had to find a trade-off between prediction performance and model accuracy and plan to work on a more lightweight network traffic pre-

diction in the future. Our F-IDS could replace such a lightweight IDS, as all clients train their own fast and lightweight local models based on their own data over time and improve the global model collaboratively.

Railway Systems Apart from vehicular networks, railway systems became an attractive target for hackers, which heavily hit the Italian and Danish railway systems in 2022 [82, 17]. Mainly due to the open accessibility of the infrastructure and the advancing digitization of the systems, transport businesses have growing concerns about the safety and security of their infrastructure [69]. Different control systems for trains, such as the European Rail Traffic Management System[1], rely on cyber-physical systems for the operational safety. These systems are used for communication between trains, infrastructure, and control towers and are a very important part of the modern railway. Wireless communication protocols are used to transmit commands to control train movements and signals. The packets exchanged via the network contain information about signal states, stops, speed limits, and track conditions [69]. Once received, the locomotive performs appropriate actions to ensure a safe operation. Attackers in a railway network could compromise the safe navigation of trains which possibly leads to collisions between trains or other unimaginable consequences. Therefore, researchers have already developed several IDS for railway systems specialized to ensure communication security [69].

Gao et al. proposed an intrusion detection method based on machine learning and a state observer to detect and recognize various attacks in railway communication [38]. Their system detects anomalies in the wireless network data and observes the train's physical state. The proposed system includes two layers, the first layer to detect and identify wireless network attacks based on machine learning algorithms, and the second layer to detect the abnormal physical state. Since existing datasets were not sufficient for the training and evaluation, the authors collected data from a railway simulation platform and added DoS and Data Spoofing attacks. By combining the results of the two layers, the authors reported an improved intrusion detection result in detecting anomalies [38].

Kour et al. proposed a prediction and detection technology called the Railway Defender Kill Chain, used to predict, prevent, detect, and respond to ongoing cyberattacks [57]. Their system integrates and combines multiple existing technologies and methodologies to detect and respond to cyberattacks. An action matrix is used to specify the countermeasures against ongoing attacks to minimize loss and unavailability. The authors suggest implementing more proactive cybersecurity approaches, where a defender can detect cyberattacks in time and mitigate them by implementing the right defense strategy, rather than employing forensic teams to restore the systems after a successful attack and potential exploitation. In the future, the proposed system should be applied by cooperating railway companies.

In conclusion, the biggest problems of recently developed IDS are the limited effectiveness against novel attacks, problems due to the restricted processing and memory capacity, and difficulties with the real-time and continuous data stream processing [43]. Although some good local IDS solutions are available, these systems still need to be further developed and improved. Especially the limited effectiveness against unknown attacks and the insufficient adaptability to different systems are major challenges. Additionally, the real-time evaluation of the network data is a problem of current IDS. In the future, such problems could be solved by F-IDS, which we will discuss in more detail in the following section.

[1] https://www.era.europa.eu/domains/infrastructure/european-rail-traffic-management-system-ertms_en

3.2 Federated Learning in Intrusion Detection

To implement a generic IDS for complex transportation systems containing various vehicles and widespread infrastructure, federated learning (FL) could be used. As described in chapter 2, FL is a machine learning paradigm where a global model is learned across distributed devices without sharing the original training data, resulting in a significantly increased speed, real-time processing capability, and security.

In recent years, FL has benefited from a wide range of applications like word prediction [68] and federated visual object detection [64]. Furthermore, researchers already combined FL mechanisms and IDS, for example, to detect intrusions in industrial cyber physical systems (I-CPS) [62]. Large industrial infrastructures with intelligent networks using 5G communication, software-defined networking, and artificial intelligence have an increased likelihood of being exposed to attacks, e.g., as it happened in 2021 to the US fuel and gas pipeline system [86]. This attack affected one of the largest oil pipelines in the USA and led to the pipeline being taken off the grid, causing supply bottlenecks and delivery failures [21]. However, securing these large-scale, complex, and heterogeneous I-CPS is extremely challenging due to the insufficiency of high-quality attack examples. Li et al. developed a F-IDS using a convolutional neural network to detect cyber threats in I-CPS [62]. The proposed deep learning scheme called DeepFed uses a novel FL framework for the effective detection of various types of cyber threats in a privacy-preserving way. A secure communication protocol was designed to preserve the interception of model parameters during the transmission in the training process. The authors conducted extensive experiments on real data from an industrial gas pipe lining system and reported a high effectiveness. With seven clients and after ten communication rounds, the model achieves an accuracy of 99.2%.

Another recently proposed field of application for FL in intrusion detection are medical cyber physical systems (M-CPS). These large networked systems of medical devices deliver high-quality care by allowing continuous monitoring and treatment of patients. In most cases, such a system stores sensitive medical data and personal health data that should be secured against cyberattacks. To overcome the problems with the heterogeneity of the devices (such as body sensor nodes and smartwatches), Schneble et al. proposed a new F-IDS [89]. At the same time, the communication, and computation costs involved in traditional machine learning were minimized by using FL. The system was evaluated with real patient data together with up to eight clients and tested against different sorts of security attacks such as denial of service, data modification, and data injection. The metrics were averaged over five communication rounds and the proposed F-IDS achieves a detection accuracy of 99.0% and a false positive rate of 1.0% along with a reduced network communication overhead. Even without using all client weights in each training round, but only specific subsets to reduce communication and energy costs, there was no trade-off in detection accuracy. These results indicate that even autonomous transportation systems could benefit from such a system by an improved accuracy and a decreased number of false alarms.

In the process of digitalization, there are more and more internet of things (IoT) devices, entering everyday life and connecting via the internet. In 2022, more than 17 billion IoT devices were spread all over the world, however, these devices are often vulnerable to cyberattacks due to an insecure design, implementation, and configuration [66]. As a result, many networks already have vulnerable IoT devices that could easily be compromised. Since there are hundreds of very heterogeneous devices on the market, it is necessary to use FL to train a precise de-

tection model. Nguyen et al. developed a F-IDS for IoT devices without the need for human intervention or labeled data [76]. They propose a FL approach to aggregate behavior profiles from different devices. The system was trained using data autonomously labeled with the device type and evaluated in a real-world smart home system. Long-term evaluations showed high effectiveness and efficiency, with about 95% detection rate and 0.257 seconds needed for detecting compromised devices.

In conclusion, there are already several application areas for F-IDS. Nevertheless, the use of FL in vehicles or more advanced autonomous transportation systems is still at its beginning with very few publications so far. Elbir et al. published a case study about possible applications for FL in vehicles, concluding that autonomous driving, infotainment, path planning and maybe even the car's motion controlling could benefit from FL [35]. However, the use of FL for intrusion detection still faces other challenges than those mentioned above, which we will discuss in detail below.

3.3 Challenges in Federated Learning

Recently, some great strides have been made in the development of federated learning (FL) algorithms. However, there are still challenges in the development of FL systems [68]. Two major attacks could affect a FL system: 1) poisoning attacks and 2) inference attacks. Poisoning attacks aim to prevent a model from being learned at all, or try to bias the model to produce wrong inferences, whereas inference attacks target the privacy of the data and attempt to reconstruct the original data. Given that in FL only the trained models, weights, or gradients are transmitted, it is difficult to infer the original sensitive training data, but not impossible [37]. Even from a small number of model weights, attackers could extract information about clients' data [79]. In [68], McMahan et al. differentiate between insider and outsider attacks. To diminish model accuracy or introduce backdoors into the global model, a malicious participant could manipulate the input from inside the system. Therefore, insider attacks are generally stronger than outsider attacks. Compared to traditional learning algorithms on separate devices, any participant could violate the FL system. One common example of a poisoning attack is the label-flipping attack, where a single client flips the labels of some data samples from one class to another, while the features of the data are kept unchanged to negatively impact the model performance [50]. Apart from cyberattacks, there are a few more challenges in the operation of FL systems. Problems like data heterogeneity and different response times of clients are the subject of current research [114]. The heterogeneity of the client's datasets could result in a biased machine-learning model. Furthermore, response times or the number of data samples may differ from client to client due to different hardware or network connectivity [103]. In our contribution (chapter 4), we will discuss these issues in detail and describe their impact on the system. To minimize the risk of data leakage and prevent the system from being affected by the above-mentioned attack methods, different encryption methods have been proposed by researchers. The two following methods to protect the model during transmission became very popular over the last years [110]:

Differential Privacy Some researchers use differential privacy in FL for data privacy protection [33]. In differential privacy, noise is added to the data, or generalization methods are used to hide certain sensitive attributes until attackers cannot distinguish between the individual data and the noise anymore [110]. This, however, could lead to weaker model performance and accuracy [63].

Homomorphic Encryption Homomorphic encryption is used to protect data using various encryption methods [84]. Unlike differential privacy protection, the data cannot be guessed by third parties, since the data is encrypted before transmission and decrypted after the recipient has received the data. This makes the transfer process much safer, which is why we have implemented this concept in our system architecture.

4

Contribution

In this chapter, we will describe our developed system, the global model, and the communication between clients and server in depth, as well as necessary implementation details. We analyze the advantages and disadvantages of synchronous and asynchronous federated learning (FL) methods and explain their suitability for our use case. We implement the federated averaging (FedAVG) approach together with the necessary communication between clients and server and a high-level safety message encryption. Furthermore, we will give reasons for our selected datasets and parameters. The implemented code can be found on Github[1].

4.1 Architecture for a Federated Intrusion Detection System

There are various frameworks and libraries available that can be used to develop FL applications. Published frameworks can be categorized according to their main objectives [15]: **Simulation-oriented** frameworks provide multiple development and benchmark tools, and focus on adding new functionalities and evaluation purposes (e.g., using emulation and virtual devices). In contrast, **production-oriented** libraries provide enterprise-level solutions, giving support to various FL scenarios, by focusing on usability and productivity. Most libraries support FL centralized algorithms as federated averaging (FedAVG). In our first approaches for the implementation of the federated intrusion detection system (F-IDS), we wanted to use the simulation-oriented TensorFlow framework[2]. Unfortunately, many frameworks have limitations, i.e., they cannot be deployed to multiple client devices, are not publicly available or cannot be configured properly. Therefore, we decided to move away from pre-produced frameworks towards a standalone implementation. Using our own system would allow us unlimited modifications and improvements, the client programs could be optimized for execution inside the BeIntelli infrastructure, and we always have total control over all parts of the FL algorithm. For our F-IDS, we adapted the most popular FL algorithm FedAVG, introduced by McMahan et al. in [67], for the application in autonomous transportation systems.

[1] https://github.com/zunzer/federated_intrusion_detection

[2] https://www.tensorflow.org/federated

4.2 Training and Prediction

Due to the lack of training data with labels in real-world applications, it's inevitable to use an unsupervised machine learning technique. Autoencoders are artificial neural networks that are specialized for anomaly detection without requiring labeled training data. They are often used for dimensionality reduction or compression, where the input is the same as the output [32]. Apart from dimensionality reduction, they are widely used in image compression or denoising (e.g., for facial reconstruction [78]), feature extraction (e.g., for facial expression recognition [48]) or, most importantly, for anomaly detection. A major advantage for autoencoders in the field of cybersecurity is that there is no need for labeled data in the training phase, which is why even larger companies like NVIDIA decided to use multiple autoencoders for intrusion detection [8]. Simplified, an autoencoder compresses the input into a lower-dimensional representation and then tries to reconstruct the output. For this, common autoencoder implementations contain two main components: The encoder and the decoder. The encoder compresses the input and produces the prediction vector while the decoder reconstructs the input using this prediction vector (see Figure 4.1). The output of the decoder will not be the same as the input, in the best case it will be close [32]. In each iteration of the training, the model reconstructs the input data using this reconstruction vector and tries to change the model weights to minimize the difference between the input and reconstructed output. Until now, various optimized autoencoders have been proposed by researchers with certain advantages and disadvantages [81]. However, we will limit ourselves to the basic methods that are sufficient for our system. In the following, we will take a closer look at one of many mathematical definitions of an autoencoder as described in [93].

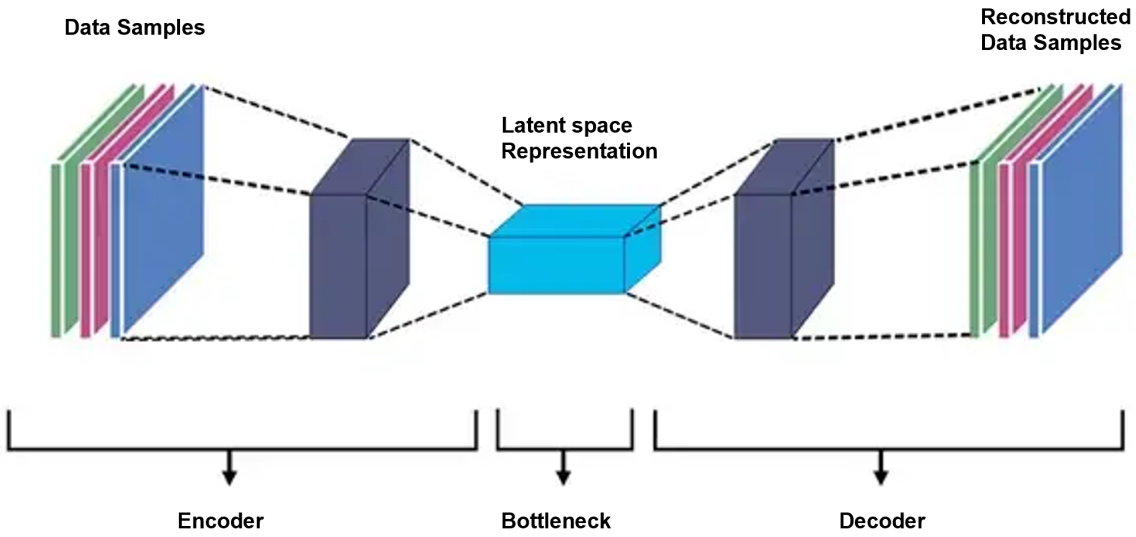


Figure 4.1: Structure of an autoencoder [2]

In an autoencoder, the encoder function can be denoted by E_ϕ. This function E_ϕ maps the original data X to the latent space representation F. The decoder function $D_\psi(x)$ maps the latent space representation F at the bottleneck to the output X, the same size as the input:

$$E_\phi : X \to F \tag{4.1}$$
$$D_\psi : F \to X \tag{4.2}$$

Usually, both the encoder and the decoder are defined as standard neural network functions, e.g., a one-layer encoder and decoder can be defined as:

$$E_\phi(x) = \sigma(Wx+b) \tag{4.3}$$
$$D_\psi(x) = \sigma'(W'x+b') \tag{4.4}$$

Here, σ is an activation function, W is the weight matrix that we later transmit between clients and servers, and b is a bias vector. The activation function can be used to map the model's weights to a specific range of values to prevent them from becoming too small or too large. Determining the best activation function suitable for a specific application is part of the hyper-parameter tuning and can be done in future advancements to further improve the performance of our system. Additionally, we need to define $\Delta(x,x')$ as a function to measure the difference between input x and the reconstructed output x'. With those equations, we can now create an example loss function for the autoencoder.

$$L(\phi,\psi) = ||\Delta(x, D_\psi(E_\phi(x))||^2 \tag{4.5}$$

This loss function is the evaluation metric to judge the performance of a model and is used for training a neural network. We use the mean squared error (MSE) as it is a common loss function for machine learning [28], and especially for similar intrusion detection tasks with autoencoders [61]. In the training, the algorithm is searching for the optimal weights minimizing the overall loss by mathematical optimization techniques. The optimizer is necessary for finding the model weights with the lowest loss and could be e.g., stochastic gradient decent (SGD), Resilient Back Propagation, or adaptive moment (Adam). As proposed in [61] for intrusion detection tasks with autoencoders, we will use Adam as an optimizer. Adam is an optimization algorithm that achieves good results fast, as it was a further extension of SGD to update network weights using an adaptive learning rate [18]. All the specified parameters are variably changeable and can be adjusted later if needed.

For the implementation of the autoencoder itself, we use the TensorFlow Keras API[3] in Python. After importing all necessary libraries, like Pandas[4] for data handling, NumPy[5] for mathematical operations, and last but not least the TensorFlow[6] library, we create the autoencoder. This model consists of an input layer, the encoder layers, and the decoder layers. In the input layer, we specified the shape of the dataset. In the case of the by-us used evaluation dataset, we have an input shape of 79. In [100], the authors proposed to use input and output layers of size n where n is the number of data features for the input data, and three hidden layers of sizes $\frac{n}{2}$, $\frac{n}{4}$ and $\frac{n}{2}$. If we use too small dimensions for the hidden layers, the ability to

[3]https://www.tensorflow.org/tutorials/generative/autoencoder
[4]https://pandas.pydata.org/
[5]https://numpy.org/
[6]https://www.tensorflow.org/

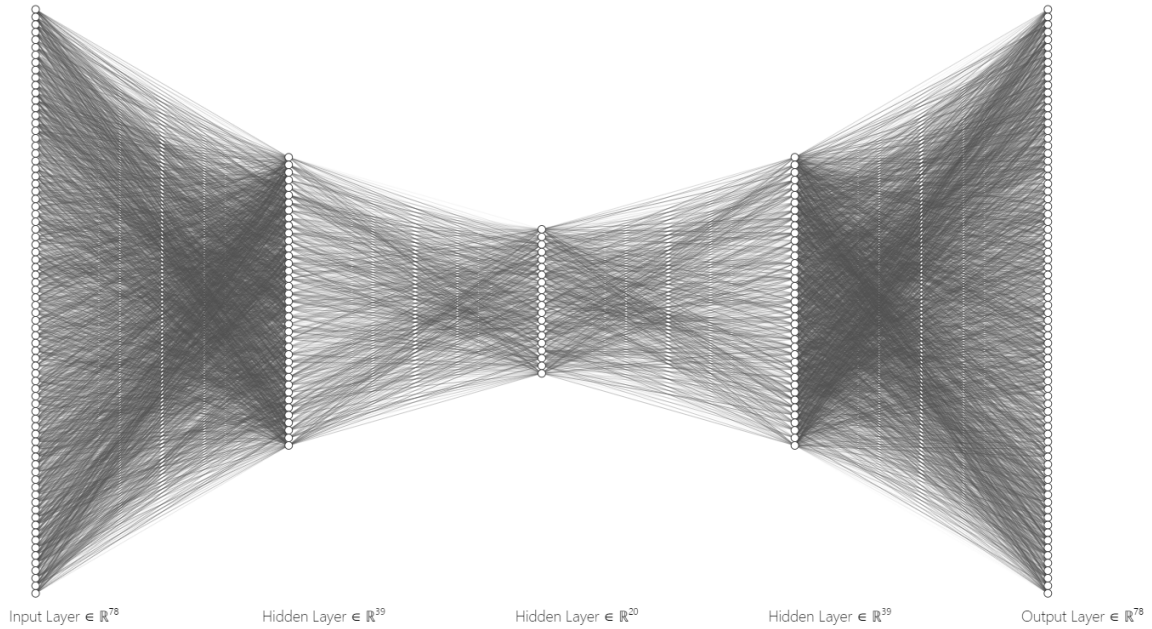


Figure 4.2: Visualization of the autoencoder with input and output size $n = 78$ suitable for the used evaluation dataset and three hidden layers of sizes $\frac{n}{2}$, $\frac{n}{4}$ and $\frac{n}{2}$

regenerate the original data is limited, but if we use too large dimensions, there is no feature extraction, and the compression will have no meaning. We visualized the implemented autoencoder, which consists of input and output layers based on the evaluation dataset in Figure 4.2. A problem of many machine learning algorithms is called overfitting [111]. This happens when the model does not generalize well from observed data to unseen data, so the algorithm cannot accurately predict the labels of unseen data. To avoid this problem, there are different mitigation techniques [111], e.g., a dropout rate could be added to randomly drop neurons of the autoencoder during training, so that the model always has a slightly different structure. The model becomes less likely to overfit, and it also prevents the algorithm from memorizing noises. In the evaluation, we will analyze if our algorithm tends to overfit or if it is able to learn correctly.

Because most modern network applications, particularly autonomous transportation systems, have a high data throughput and a high detection rate, a F-IDS must be capable of handling this data in prediction and have a correspondingly high throughput rate. As described in chapter 2, in offline prediction, the training data is collected over time and stored in static datasets. Because this method is too slow for our purposes and we want to detect anomalies immediately, we have to use online prediction to keep up with the large data streams. In this case, incoming network traffic is analyzed and filtered for malicious traffic directly after transmission before attackers can reach critical components of the system. When the autoencoder has been trained and new data points arrive, we use online prediction in mini-batches of size 32 to identify the anomalies which are characterized by a high reconstruction error. For each input vector from the incoming data batch, we simply calculate the difference between the reconstructed data and the original input data. The outliers have a higher reconstruction error because they differ from the regular data. In research, different methods are used to define at which deviation a data point should be considered as an outlier.

Static Prediction Loss: The static prediction loss assumes that exactly e.g., 1% of the most deviant traffic is malicious. We thus use a static percentage which was determined in advance, whereby an always equal proportion of data points is selected as anomalies. Since we expect a high amount of normal traffic and a variable number of anomalies, this approach is not really suitable for autonomous transportation systems and real-world applications.

Median Absolute Deviation about the Median To have a better-adopting autoencoder, we use the median absolute deviation (MAD) presented in [46]. This is a robust measure of dispersion in descriptive statistics and indicates how far a sample deviates from the median on average. With this method, we consider points as outliers depending on how much they deviate from the median. The mean cannot be used in this case because it can be strongly influenced by outliers. For a dataset D containing the data samples $x_1, x_2, x_3 ... x_n$ we calculate $\tilde{x}$, which is simply the median of the data. Then we calculate the absolute difference between each point and the median. Afterward, we have to calculate the median again to get the MAD:

$$MAD = median\{|x_i - \tilde{x}|\} \tag{4.6}$$

With the median of the data distribution and the calculated values for MAD, we can calculate the modified Z-scores (M_i) which are defined as follows [46, 98].

$$M_i = \frac{0.6745(x_i - \tilde{x})}{MAD} \tag{4.7}$$

Data samples will be labeled as outliers when $|Mi| > k$, where the result can be optimized by adopting parameter k for the application area. In chapter 5 we will determine the best value for the threshold k based on our evaluation dataset.

As described in chapter 2, a distinction is made between synchronous and asynchronous FL. Now that we have introduced the global model, the training, and the prediction, we want to review both approaches, explain the differences in suitability for a F-IDS, and present the mathematical idea behind the most popular aggregation algorithm FedAVG.

4.3 Synchronous & Asynchronous Federated Averaging

Our goal was to implement a F-IDS for autonomous transportation systems. After we developed the global model for detecting outliers, we need to implement the server-side model aggregation and the data exchange. In FL, we usually differentiate between synchronous and asynchronous learning, even though there are various further developments proposed, such as semi-synchronous learning [94]. We will limit ourselves to the two most common categories synchronous and asynchronous learning, and describe the principles based on the FedAVG algorithm already presented in chapter 2. Usually, synchronous FedAVG contains several steps for training the global model [108]: First, the algorithm starts with the initialization of the global model and necessary parameters. For this, the aggregation server prepares the initial global model w_G^0 and configures the training settings, such as loss function and optimizer. The aggregation server sends the global model weights w_G^0 to the first client c_1 of all n clients (denoted as C), where this client starts its own local model training and retrieves the new local model weights. These weights are transmitted to the server, where they are stored for aggregation after the round is completed. Let t be the current iteration number. Based on the global

model w_G^t in iteration t, each client trains its local model and obtains the local models w_c^t, where $c \in [1, C]$ one after another. The local models w_c^t are sent back to the aggregation server and are collected for the aggregation process after receiving all local model weights. In the last step, the global model aggregation is conducted, where the server aggregates the local models and generates a new global model by summing up all received local model weights and dividing through the number of clients:

$$w_G^{t+1} = \frac{1}{C} \sum_{c=1}^{C} w_c^t \tag{4.8}$$

After that, the aggregated global model w_G^{t+1} is sent back to the first client for the next iteration of federated training.

After the implementation of synchronous training, we quickly encountered the first performance problems caused by long training and long response times. Due to differences in computation and communication capabilities among the heterogeneous devices, synchronous FL has incredibly high performance losses. Even with a small number of clients ($C < 5$), the sequenced training takes a long time when the server waits for each client in turn to complete the training. Hence, asynchronous training was necessary to reduce the time costs and to decrease the influence of a slow or broken client, and therefore improve the overall systems' efficiency. In asynchronous FedAVG, the global model aggregation happens whenever a new local model is received by the aggregation server instead of waiting for all clients to complete. The major steps of the asynchronous FL algorithm are as follows [108]: Again, the algorithm starts with the initialization. Then, the aggregation server broadcasts the global model w_G^0, but this time to all clients C at the same time. The clients start to train their local models based on the most recent global model obtained from the aggregation server. When a client i finishes the training of the local models $\{w_i^t\}$, the weights are returned to the server, where the global model aggregation is conducted. The newly collected local model is aggregated with the latest global model by averaging the weights as follows:

$$w_G^{t+c} = \frac{w_G^{t+c-1} + w_c^{t+c}}{2} \tag{4.9}$$

After that, w_G^{t+k} is the new aggregated global model and will be used for the next training round. The immediate model aggregation in the asynchronous FL significantly reduces the waiting time and thereby improves the efficiency [65]. To prevent slow clients from delaying the process and lowering the total performance, timeouts for each round can ensure that clients are removed from the training process when they are not answering. After we implemented this asynchronous FedAVG algorithm and specified the basic properties of the system, we need to implement the message exchange. For this, we developed a simple, customized protocol for the message exchange between clients and server and implemented message encryption, which we will describe in the following section.

4.4 Encryption

As described in chapter 3, it is necessary to encrypt the traffic between clients and server to avoid that sensitive information is used by unauthorized persons. For the encryption between client and server, we decided to use one of the most common encryption methods, called the

(a) encrypted package

```
No.  Time        Source       Destination    Protocol    Length   Info
  4  0.008103    127.0.0.1    127.0.0.1      TCP         77       54261 → 65432 [PSH, ACK] Seq=1 Ack=1 Len=33

Frame 4: 77 bytes on wire 616 bits, 77 bytes captured 616 bits on interface \Device\NPF_Loopback, id 0
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 54261, Dst Port: 65432, Seq: 1, Ack: 1, Len: 33
Data 33 bytes

0000  d6 1c 52 25 68 80 a9 b7 4e 2c 0b 3c 68 16 2b 8e   ..R%h...N,.<h.+.
0010  8f b2 3c 80 fb 89 30 67 89 00 64 72 fe 8d 76 89   ..<...0g..dr..v.
0020  2e                                               .
    Data: d61c52256880a9b74e2c0b3c68162b8e8fb23c80fb89306789006472fe8d76892e
    [Length: 33]
```

(b) unencrypted package

```
No.  Time        Source       Destination    Protocol    Length   Info
 10  10 13.9820  127.0.0.1    127.0.0.1      TCP         77       54261 → 65432 [PSH, ACK] Seq=1 Ack=1 Len=33

Frame 10: 77 bytes on wire 616 bits, 77 bytes captured 616 bits on interface \Device\NPF_Loopback, id 0
Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1
Transmission Control Protocol, Src Port: 54224, Dst Port: 65432, Seq: 1, Ack: 1, Len: 33
Data 33 bytes

0000  5b 22 72 65 67 69 73 74 72 61 74 69 6f 6e 22 2c   ["registration",
0010  20 37 2c 20 36 35 34 33 33 2c 20 36 35 34 33 32    7, 65433, 65432
0020  5d                                                ]
    Data: 5b22726567697374726174696f6e222c20372c20363534333332c20363534333325d
    [Length: 33]
```

Figure 4.3: In contrast to encrypted communication (a), unencrypted communication (b) reveals sensitive information in the payload to every attacker in the network

advanced encryption standard (AES). The algorithm was developed by Joan Daemen and Vincent Rijmen under the name Rijndael [29] and has been established in the year 2000 as the American standard for encryption. AES is a symmetric encryption method where the encryption and decryption keys are identical. The algorithm offers a very high level of security and is even used by governments and the US military [83]. The AES encryption algorithm specifies numerous transformations that must be performed in multiple rounds on the data to encrypt them. Briefly summarized, the data is split into equal blocks (e.g., 128, 256) and the three cipher transformations i.e., the substitution of data using a substitution table, row shifting, and column mixing, are repeated over several rounds on these blocks to obtain the encrypted or decrypted data. For more details, see [29].

4.5 Client- & Server Communication

We adopted the FedAVG algorithm for one server and multiple clients. To exchange model weights, we needed to implement the communication between all parties by hand, using the python socket library[7] and a custom protocol (see Figure 4.4). In the beginning, all participants and the server bind to a specific port. To register for training, the clients have to send a special registration message together with meta information like port and IP address to the server. When receiving this message, the server adds the client to the training participants list. As we have implemented asynchronous FedAVG, in each iteration, the server broadcasts the global model weights to all registered clients, which immediately start the training process. After a client trained its local model, the new weights are sent back to the server for the aggregation process. In addition, the client stores all values required for the evaluation since the last con-

[7] https://docs.python.org/3/library/socket.html

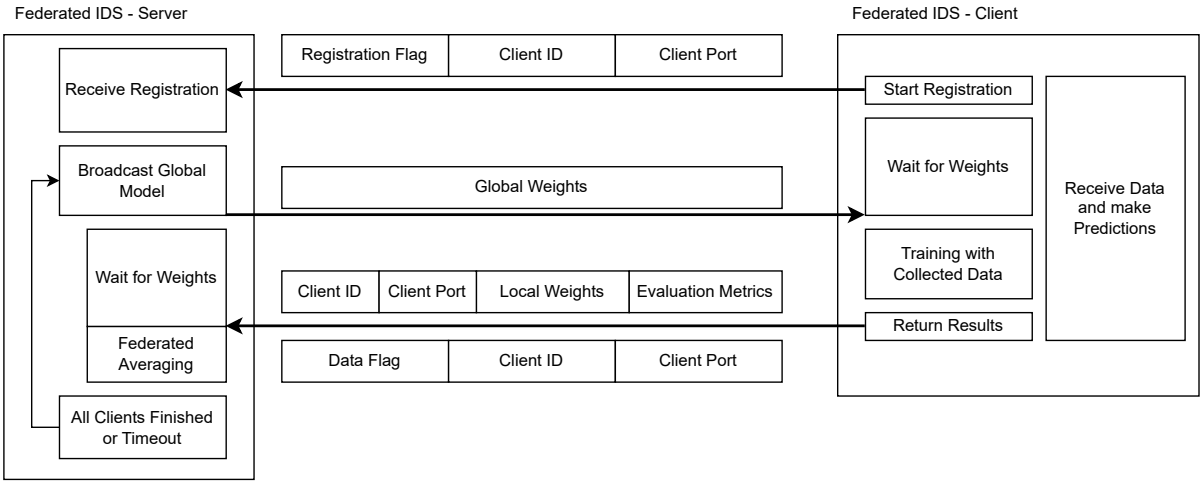


Figure 4.4: Message exchange between client and server

nection to the server and adds them behind the response to the server. It could happen that a client does not have enough data collected for participating in the current training round. Then, the response of the client will contain a specific no-data flag instead of global weights so the server skips the aggregation process for this client. To avoid slow clients delaying the process and decreasing the overall performance, we set timeouts for each round, after which clients are removed from the training process when they are not responding. Simply removing not responding clients could lead to a biased model where only the fast-responding clients participate in training. In the future, we are planning to implement other methods, to deal with not responding clients, e.g., increasing their influence during the aggregation. In addition, the client starts a new registration request if it has not received any global weights from the server for too long.

4.6 Client Dataset Generation and Preprocessing

For testing and evaluation purposes, we started using a dataset that was created for the Third International Knowledge Discovery and Data Mining Tools Competition (KDD(Cup)'99)[8]. The task was to build a network intrusion detection system (IDS), containing a method capable of distinguishing between attacks and normal connections. This dataset contains a wide variety of intrusions simulated in a military network and labeled as normal or with the type of attack. Before training, the dataset was loaded, one-hot encoded, randomized, separated into 10 client datasets, and saved to CSV files. One-hot encoding is a common method in machine learning for dealing with categorical data, such as protocol types, because many machine learning models require numeric input variables. One-hot encoding is used to transform categorical variables into numeric values by removing categorical variables like Protocol type and replacing them with binary variables for each value, for example TCP or UDP [20]. We used randomization to keep the ratio of attacks and normal traffic equal for all clients and to prevent a client gets assigned too many attacks. After conducting the first tests, we encountered a big problem. In the KDD(Cup)'99 dataset, the ratio of normal and malicious traffic is too bad for training

[8] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

our autoencoder with it. Unfortunately, the dataset is very unrealistic by now for real-world applications [30], so we decided to switch to the newer CIC-IDS17 dataset[9] to get more recent and reliable data. The CIC-IDS17 dataset includes benign and the most recent widespread attacks and a lot more real-world connection data than the KDD(Cup)'99. The authors created a network containing routers, firewalls, switches, and devices with different versions of operating systems [90]. The data were collected on 5 consecutive days from Monday to Friday in a network using a set of publicly available test tools to simulate the attacks. Besides some others, it contains distributed denial of service (DDoS), brute force, injection, infiltration, and botnet attacks [90], a detailed breakdown can be found in chapter 5. The few values that weren't given in the CIC-IDS17 dataset were replaced with 0. This dataset better reflects the current network environment, particularly as it simulates attacks from the Internet. Again, we split the CIC-IDS17 dataset into datasets for 10 different clients. With the help of this well-suited dataset, we started the evaluation tests on the following experimental setup.

4.7 Experimental Setup

To create a system that is as close as possible to the real-world scenario in the BeIntelli infrastructure where a road-side unit (RSU) receives the data at one specific port, we had to create a data delivery simulator. This simulator frequently sends out traffic loaded from CSV or PCAP files to one connected client via the data delivery socket. The client collects the incoming data samples and saves them into small mini-batches. These small mini-batches are analyzed for anomalies and then handed to the training process for further model training (see Figure 4.5). For the deployment, it is necessary to have multiple threads running on the client. One thread in the client system is receiving the incoming data, creates the data containers, and starts the prediction. After the prediction, the data samples are handed to the training thread, where the model training takes place.

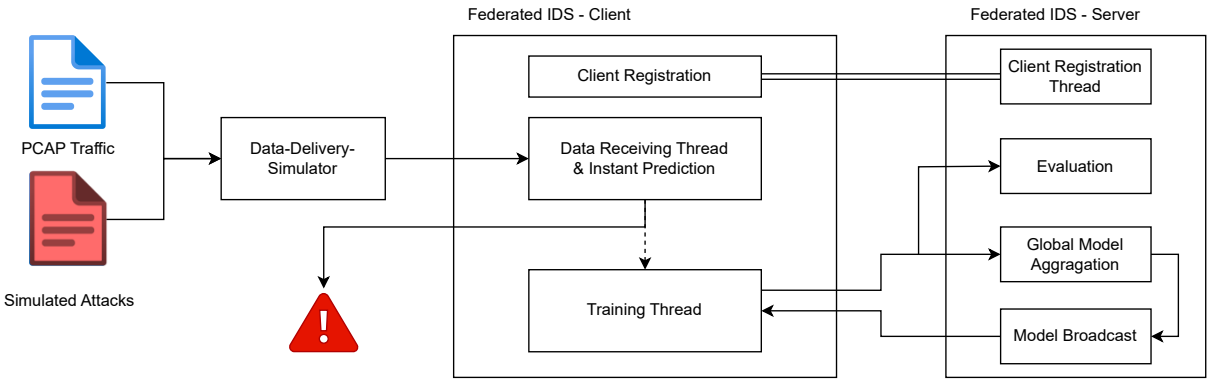


Figure 4.5: Overall architecture and processes of the implemented F-IDS

When implementing the threads for the federated client processing, we ran across a few issues. Initially, we had two threads on the client: One for prediction and training and another one only for the processing of the incoming data. Because no incoming data could be predicted

[9] https://www.unb.ca/cic/datasets/ids-2017.html

during training in this setting, the time between receiving an anomaly and reporting it as malicious became very high. We wanted to improve this detection time by implementing a third thread only for the prediction, unfortunately not without other drawbacks. We encountered a rapidly decreasing data receiving performance (see Table 4.1), most likely due to a greatly increased computational effort. This led us to our current solution, in which we merged training and prediction on one thread while combining data receiving and training in another. This ensures a large volume of processed data together with a rapid detection rate on the clients. The server contains two threads too, one is just for the receiving of new client registrations, and the other does the main FL tasks. In the following chapter, we will describe the setting inside the BeIntelli infrastructure and the collection of real-world data.

			Received Data samples in seconds		
Thread 1	Thread 2	Thread 3	1000	5000	Detection time:
Prediction + Training	Data receiving	-	1.2052s	10.1262s	Slow
Prediction	Training	Data receiving	43.4499s	190.7817s	Medium
Prediction Data receiving	Training	-	6.4306s	33.7751s	Fast

Table 4.1: Comparison of the processing speed of different thread installations for 1000 and 5000 data samples

4.8 Deployment to BeIntelli

The autonomous transportation system operated by the BeIntelli project contains different types of devices, that communicate with each other using a wide variety of transmission protocols. The road infrastructure is equipped with road-side units (RSUs) from the CohdaWireless company, a leading supplier of innovative vehicle connectivity solutions[10]. The software running on these RSUs allows accurate vehicle positioning for autonomous vehicles in smart cities. For the collection of the real-world data, we used two Cohda Wireless RSUs MK5 running in the BeIntelli Infrastructure. These MK5s provide the latest connectivity with new chipsets, support for 2G, 3G, and 4G, and are equipped with Wi-Fi and Bluetooth to ensure good short-range communication. The newer version MK6 is already prepared for 5G, IP67 weatherproof, and one single box can process the entire data exchange at a complete intersection. RSUs perfectly fit in a federated approach since they are widely distributed in the test environment and capable of collecting and processing data. FL would be applied across all RSUs to realize the collective use of all data for training a machine learning model with it. An enhanced IDS could then be deployed to all RSUs which otherwise wouldn't have an adequate amount of data to train a local model on their own. The autonomous vehicles driving in the system are equipped with on-board units (OBUs) and can be part of the FL system too. To be prepared for cyberattacks in the future, it is necessary to implement an IDS for such autonomous transportation systems that can process the accruing network data. Furthermore, FL allows us to collect data from different devices in the network, performing model training and creating a precise global model,

[10]https://www.cohdawireless.com/

Type of Anomaly	Cohda Box	Timestamp	IP addresses	Protocol Types
Attack Tool Installation	5	12:34:17 - 12:40:28	192.168.213.86 and 185.x.x.x	TCP, HTTP
SSH Brute Force	5	12:49:04 - 13:23:16	192.168.213.86 and 192.168.230.3	TCP, SSH
SSH Brute Force Response	2	12:49:04 - 13:23:16	192.168.230.3 and 130.149.98.119	TCP, SSH
SSH Privilege Escalation	5	13:25:27 - 13:31:11	192.168.213.86 and 192.168.230.3	TCP, SSH
SSH Data Leakage	2	13:25:27 - 13:31:11	192.168.230.3 and 130.149.98.119	TCP, SSH

Table 4.2: Anomalies in the real-world environment and the clients that were affected together with the timestamps, IP addresses, and protocol types

without exchanging all the background data. The F-IDS could be used to monitor intrusions in the network in real-time and supervise vehicles while moving in real-world settings.

The real-world data used for the evaluation was collected on two RSU Cohda Boxes MK5 running directly inside the BeIntelli infrastructure. One of them is located on top of the main building of the Technical University of Berlin (RSU Nr. 2: 192.168.230.3), and the other one is close to the Großer Stern (Great Star) in Berlin (RSU Nr. 5: 192.168.213.86). We received a wide variety of messages, among others: MAP Extended Messages (MAPEM, map data), Signal Phase And Timing Extended Messages (SPATEM, traffic light data), Infrastructure to Vehicle Information Messages (IVI, information concerning the vehicle), Decentralized Environmental Notification Messages (DENM, environmental information), and messages containing Geo-Networking data (GN) that could also be used to detect GPS spoofing attacks in future developments.

The dataset was created with tcpdump[11], which directly runs on the Cohda Boxes and captures all interfaces. In our test scenario, an attacker has gained physical access to Cohda Box 5 and tries to crack connected systems like Cohda Box 2 using Patator[12], a tool for multi-purpose brute-forcing and NMAP[13]. First of all the attacker installed the attack tools on Codha Box 5 (Attack Tool Installation) and tried to scan for unprotected ports in the network and to figure out passwords. After successfully brute-forcing SSH credentials for the Cohda Box 2 using an online available list of common password and user name combinations[14] (SSH Brute Force + SSH Brute Force Response), the attacker used his privileges to steal data from the Cohda Box 2 (SSH Privilege Escalation + SSH Data Leakage). A detailed description of the anomalies, the timestamps, the client that was affected and the used protocols and IP addresses can be found in Table 4.2. In our test set with an insecure password username combination, it took the attacker around 4000 attempts to figure out the correct credentials and to successfully connect via SSH. In our experimental setup, the attacker only extracted sensitive information from Cohda Box 2. In a real situation, the attacker could have launched further attacks from this point to manipulate the RSU or to compromise other devices in the network.

[11] https://www.tcpdump.org/
[12] https://www.kali.org/tools/patator/
[13] https://nmap.org/
[14] https://github.com/danielmiessler/SecLists/tree/master/Passwords

The collected data is preprocessed by converting IP addresses to integers using the python IP address manipulation libriray[15]. Hexadecimal numbers were converted to floats, the timestamps were broken down into their individual components (year, month, day, hour, minute, second), and the few remaining strings were converted to integers using feature hashing[16]. Malformed or too-short packages were skipped, so that we eventually obtained 497,927 data samples for Cohda Box 5 and 491,393 for Cohda Box 2. Each sample contains 70 features that we use for training and prediction in the federated learning setting. To avoid the autoencoder model weights getting higher or lower than the system could handle, we specified a sigmoid activation[17] function for the autoencoder. Again, the global model contained three hidden layers with size $\frac{n}{2}, \frac{n}{4}, \frac{n}{2}$ and used Adam as an optimizer and MSE as a loss function. The collected data was stored in CSV files, loaded and preprocessed by the simulation unit, and sent to two running clients. In the following, we will evaluate the results of our system on the two evaluation data sets and discuss the results.

[15] https://docs.python.org/3/library/ipaddress.html
[16] https://docs.python.org/3/library/functions.html#hash
[17] https://keras.io/api/layers/activations/

5 Evaluation

In this chapter, we will evaluate our system based on the Canadian Institute for Cybersecurity Intrusion Detection Evaluation 2017 (CIC-IDS17) dataset and with data from a real-world environment. We will start with an explanation of our evaluation metrics, followed by a detailed analysis of the performance and robustness and a critical reflection of the implemented federated intrusion detection system (F-IDS). The implemented code can be found on Github[1].

5.1 Evaluation Metrics

The evaluation of federated learning (FL) systems is more complex than in classical machine learning approaches since models are distributed over many slightly differing clients. Typically, there are two types of evaluation methods [3]: In centralized evaluation, the server has some additional data just for evaluating the updated global model, or the metrics from all clients are collected and averaged. On the other side, a client-side evaluation analyzes the performance results of one single client. We decided to try both procedures to estimate the accuracy of the overall system, and to make statements about the improvement of intrusion detection system (IDS) through federated learning. To get a more comprehensive view of the performance, we calculate the most common evaluation metrics from the confusion matrix for each client. The confusion matrix contains the following values [7]:

		Actual	
		Anomaly	**Normal**
Predicted	**Anomaly**	True Positive (TP)	False Positive (FP)
	Normal	False Negative (FN)	True Negative (TN)

Table 5.1: Confusion matrix for anomaly detection evaluation [7]

[1] https://github.com/zunzer/federated_intrusion_detection

From this, we can calculate different performance metrics. IDS are usually evaluated based on the following standard metrics [53]:

◇ True positive rate (TPR): Ratio between the number of correctly predicted attacks and the total number of attacks.

$$TPR = \frac{TP}{TP+FN} \tag{5.1}$$

◇ False positive rate (FPR): Ratio between the number of normal instances incorrectly classified as an attack and the total number of normal instances.

$$FPR = \frac{FP}{FP+TN} \tag{5.2}$$

◇ Accuracy (ACC): Percentage of all those correctly predicted instances to all instances.

$$ACC = \frac{TP+TN}{TP+TN+FP+FN} \tag{5.3}$$

For a better comparison of the results especially to existing approaches, we also want to use the F_1-Score. The F_1-Score is the harmonic mean of precision and recall and is calculated as follows [96]:

$$F_1\text{-score} = \frac{2*Precision*Recall}{Precision+Recall} = \frac{2*TP}{2*TP+FP+FN} \tag{5.4}$$

Among the metrics in the literature, the F_1-Score is the most common one. However, since true negatives are not considered, the F_1-Score could be problematic in our asymmetric scenarios where we have a lot of data points of the normal class but only a few anomalies, which is why we will view the results of the F_1-Score with caution. In our evaluation on the CIC-IDS17 dataset, we analyze the convergence of the loss function in order to make assumptions about the quality of the trained model. The goal of the model training is to minimize the loss function. Therefore, a good training process is given, when the system reaches a stable decrease of the loss function until convergence. In our system, many parameters can have an influence on the results which should be optimized before. Depending on the selection, the evaluation metrics values variate, so parameters should always be adapted for a specific use case. In the following, we measure the performance based on the CIC-IDS17 dataset. Afterward, we will show the results of the algorithm based on real-world application data.

5.2 Results for CIC-IDS2017

As described in section 4.6, the CIC-IDS17 dataset of each day was divided equally into ten subsets (e.g., Monday into $[\text{Mon}_0, ..., \text{Mon}_9]$, Tuesday into $[\text{Tue}_0, ..., \text{Tue}_9]$, etc.) and distributed to the data simulators. This ensures that each client $c_0, ..., c_{10}$ receives enough traffic from each day, c_0 would therefore learn with $[\text{Mon}_0, \text{Tue}_0, ..., \text{Fri}_0]$ and so on. Then we started ten clients, and their corresponding data simulator running on one single machine (see section 4.7). As already mentioned in chapter 4, we use adaptive moment (Adam) as an optimizer and mean squared error (MSE) as a loss function and will conduct the training over the complete dataset. Since there is an almost infinite number of combinations for fine-tuning the autoencoder, we used parameters proposed in the research literature as often as possible. To reduce fluctuations in the values and keep the diagrams clear, we had to smoothen the measured values slightly with an online data smoother[2] (smoothing factor: 0.3). We intensively tested the performance of the system to make the most crucial pre-settings in advance before collecting the final results. In the following, we will present how we determined the parameter values for epochs and the detection threshold, which we had to specify before starting the evaluation rounds.

Optimal Threshold for Detection For retrieving the optimal threshold for the detection, we measure the FPR in relation to changing values of the detection threshold on normal events on Monday as presented in [61]. Thereby we minimize the number of false positives in normal operation without any occurring anomalies while ensuring that attacks still can be detected. In Figure 5.1 we see, the FPR is decreasing rapidly until it starts to converge when reaching a MAD threshold of 70. Therefore, we selected a MAD threshold of 70 for the CIC-IDS17 dataset to minimize the FPR without having a decreased anomaly detection performance [61].

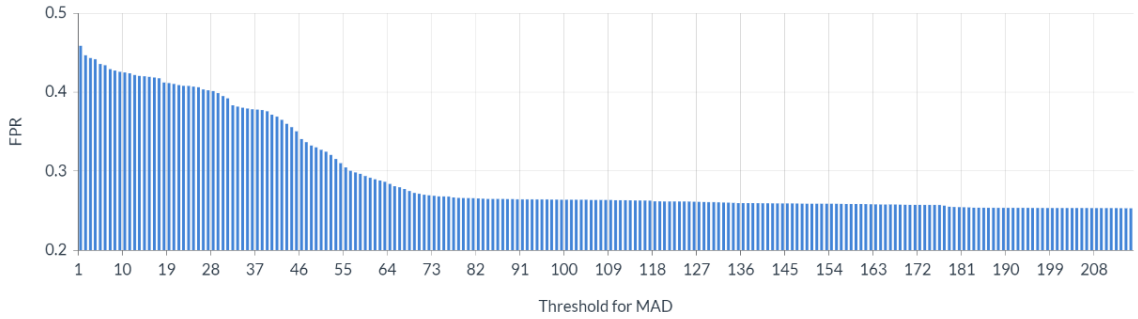


Figure 5.1: FPR measured for different MAD thresholds for normal traffic

Epochs The second parameter we specified before conducting the final tests, was the number of epochs. Since a higher number of epochs increases the time needed for training drastically, this value should be rather low. We compared the loss values of different epoch numbers for the first round of training. We decided to use three epochs for each client training round, as the loss does not decrease significantly after a larger number of epochs, and a higher number would only lead to a slowdown of the overall process.

[2] http://www.mazurka.org.uk/software/online/smoother/

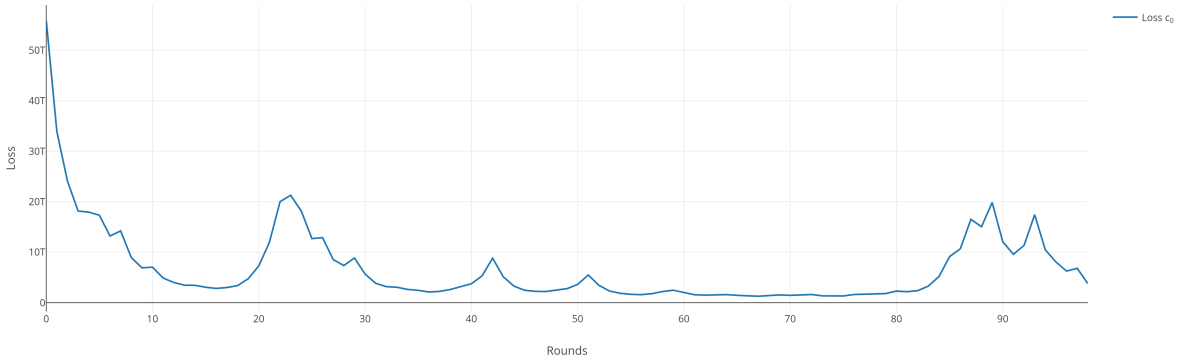


Figure 5.2: Loss for client c_0 measured in the federated test setting for the first 100 rounds

After determining these parameters for the setting, we conducted the final tests. By creating multiple processes on our test machine (AMD Ryzen 7 3700, 16GB RAM, AMD Radeon RX Vega 10) we were able to start up to 50 clients and the corresponding 50 data delivery simulators and one server. This demonstrates how effective our approach scales and how autonomous transportation systems could utilize the proposed framework without requiring excessive computation power. To run over the entire CIC-IDS17 dataset, the federated system with ten clients required about 450 communication rounds. It is important to mention, that not all clients finished at the same iteration round due to disconnections and delays of individual clients.

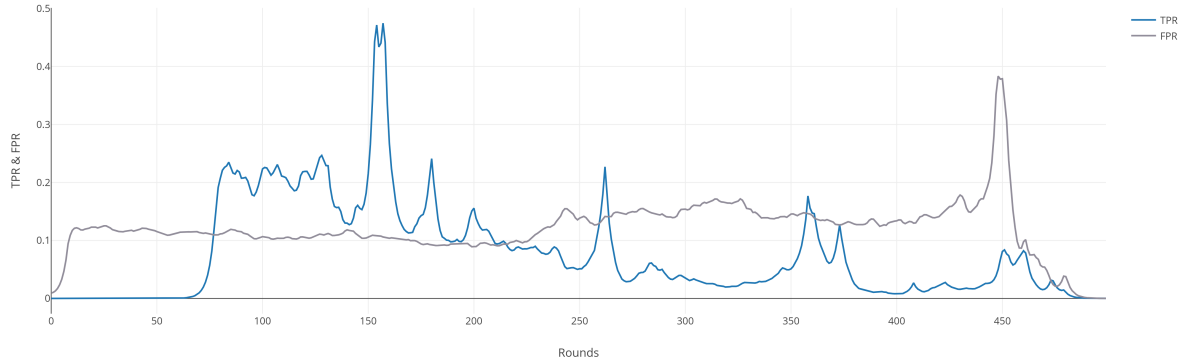
Loss Function Convergence First, we want to show that our system actually learns. Therefore, we measure the MSE loss between the original data and the reconstructed data of the autoencoder for the first client c_0 over the first 100 rounds of the training (which was mainly conducted with the normal data collected on Monday). We can clearly see that the model is exponentially getting better over the first rounds until it converges (see Figure 5.2). Subsequent peaks in the plot starting in round 80 in the training phase are caused by anomalies that have not been seen before, which increases the loss of this client for a short time. This, however, is the idea behind autoencoders, because rare anomalies have an increased loss.

Detection Time & Throughput Since the algorithm should be implemented in such a safety-critical area, we also want to analyze how fast a malicious packet is detected. Therefore, we measure the minimal, maximal, and average detection time needed between receiving an anomaly and reporting it as malicious (see Table 5.2). These values are highly dependent on a wide variety of influences, e.g., connection speed, processing power left, and the current state of the client. The detection time has much potential to be decreased in future works since simple advancements in the prediction process or the communication protocol can have a huge influence. The algorithm usually succeeds in detecting the anomalies after around 0.3 seconds. In some cases, however, the measured maximum time needed for detecting an anomaly can reach up to 20 seconds, which could be caused by connection problems or waiting times in the FL setting. The best detection time the algorithm reached in detecting a PortScan was less than 0.004 seconds. Additionally, we see that the algorithm detects a large part of the significant point anomalies very reliably. At the same time, the implemented autoencoder has clearly problems

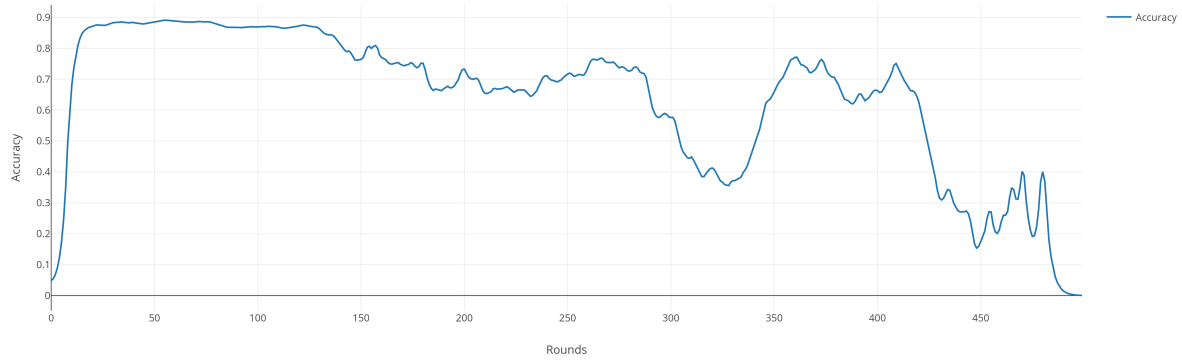
Type of Attack	minimum in seconds	maximum in seconds	average in seconds	Detected	Total	Percentage
PortScan	0.0469	18.9554	0.2616	2855	158930	1.8%
DoS Hulk	0.0818	38.0	0.4607	6251	231073	2.7%
DDoS	0.0666	19.1573	0.1884	3687	128027	2.9%
Bot	0.0882	0.5558	0.1763	80	1966	4.1%
DoS Slowhttptest	0.1029	18.4718	0.576	417	5499	7.6%
FTP-Patator	0.11	0.9267	0.2269	841	7938	10.6%
DoS slowloris	0.089	18.2582	0.4987	1308	5796	22.6%
SSH-Patator	0.0942	0.9888	0.2378	1963	5897	33.3%
Web Attack SQL Injection	0.1082	0.7588	0.3028	10	21	47.6%
Web Attack Brute Force	0.0696	1.3728	0.2028	972	1507	64.5%
Web Attack XSS	0.099	1.0822	0.2194	474	652	72.7%
Infiltration	0.1	0.8082	0.2154	30	36	83.3%
DoS GoldenEye	0.0534	9.0	0.209	8715	10293	84.7%
Heartbleed	0.111	0.2994	0.1861	11	11	100.0%

Table 5.2: Time needed by a client between receiving a sample and making a prediction in seconds. We extract minimum, maximum, and average times for each type of anomaly. Additionally, we provide the number of correctly classified attacks, the total number of this attack type inside the CIC-IDS17 dataset, and the percentage of the identified attack types

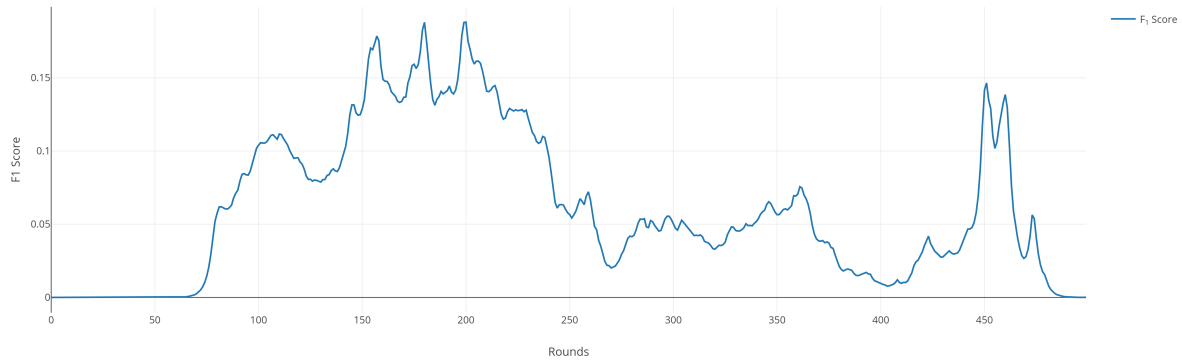
in detecting distributed denial of service (DDoS) and PortScan attacks, since these anomalies are almost equivalent to normal traffic. As described in chapter 2, DDoS belongs to collective anomalies, where a single data instance is normal by itself, but the occurrence of multiple instances can be considered as an anomaly [23]. These collective anomalies are more difficult to detect by such an IDS that examines every point by itself [5], however, DDoS attacks can be stopped relatively well by simpler methods such as firewalls [99]. Therefore, anomaly-based IDS are primarily used for detecting new and specific attacks, which can be done quite reliably as shown.



(a) FPR and TPR



(b) Accuracy

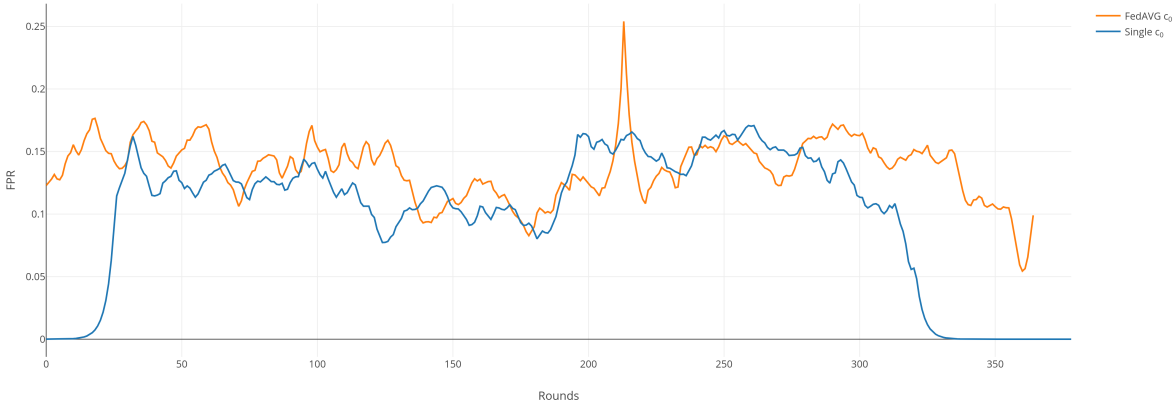


(c) F_1-Score

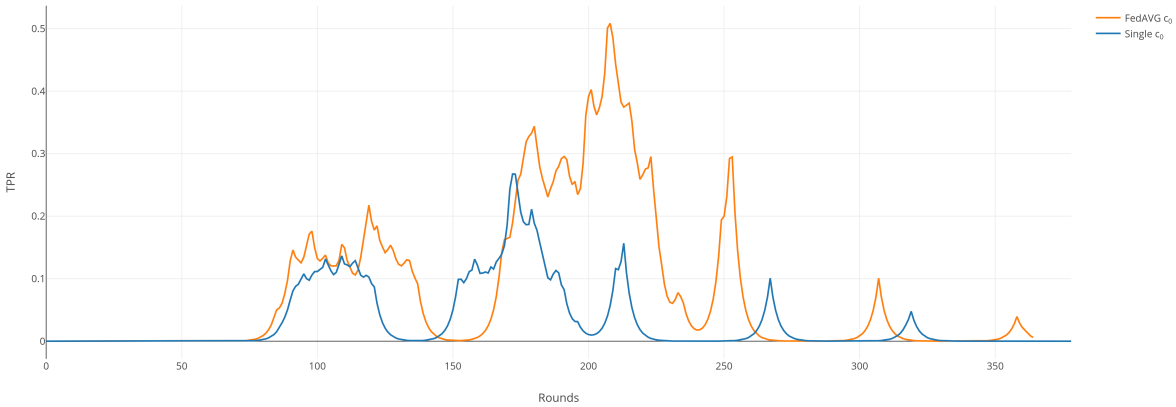
Figure 5.3: Averaged results over ten clients with the CIC-IDS17 dataset

Centralized TPR, FPR, Accuracy and F_1-Score As described in Table 5.1, anomalies are considered as the positive class. In our case, only using the TPR is not an expressive metric, due to a possibly high number of false positives. Therefore, we decided to supplement it with the FPR in Figure 5.3a. In Figure 5.3b we plot the accuracy and in Figure 5.3c the F_1-Score. Since we want to evaluate the whole system's performance, we collect the values contained in the confusion matrix on the clients and send them with the weights to the server. There, the values are averaged over all clients by summing up the values from the confusion matrice, and we calculate the corresponding metrics from them. The average FPR is constant below 20% and only increases slightly when receiving anomalies. Especially in the beginning, the accuracy is close to 90% but decreases when receiving anomalies. We see that the F_1-Score is getting better over the first 200 rounds and then starts decreasing. The F_1-Score reaches a maximum value of 0.2, which is relatively low for IDS systems but acceptable since we do not focus on fine-tuning. However, as described, it is very likely that this value is so low because our dataset is very unbalanced and contains significantly more normal traffic than anomalies.

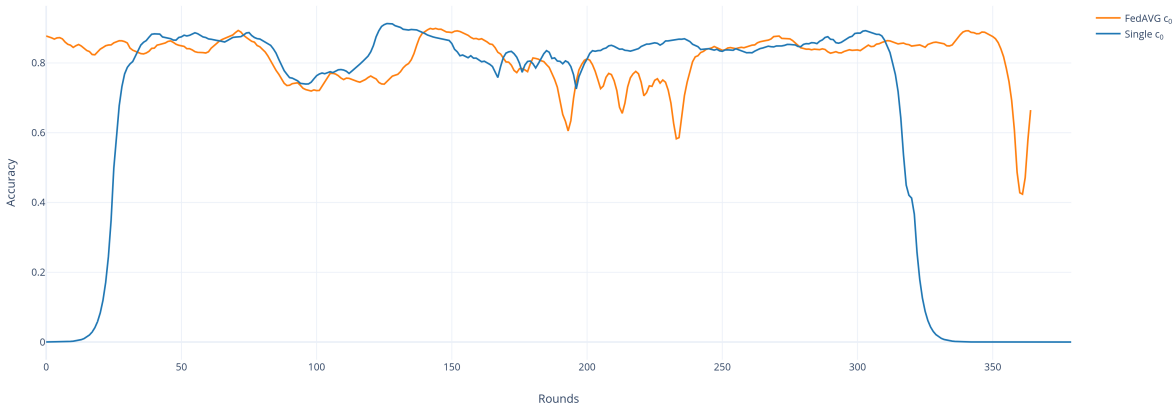
Client-side Comparison of Classical and Federated Training We also analyzed how the use of FL with FedAVG influences the performance of a single client compared to a traditional machine-learning setting. We implemented a single client, with the exact same setup as the described FedAVG setting (Adam, MSE, MAD Threshold = 70, Epochs = 3). The difference to the FL setting is, that we had only one single client c_0 which conducted the training with the data $\text{Mon}_0, \text{Tue}_0, ..., \text{Fri}_0$ without sharing data or weights with other instances, so there is no aggregation step. In a real-world application, this setup would be a client collecting its own data and training with it. We compare the results achieved by the client c_0 in the FedAVG setting, and the single client c_0 in Figure 5.4b, Figure 5.4a and Figure 5.4c. Since the FL setting is slower than a single client due to the communication overhead, the diagrams are slightly shifted and the single client finishes a few rounds before c_0 in the federated setting. We can see, that the FedAVG client has a better TPR of almost 20% when it comes to the anomalies. Over the entire period, the overall FPR and accuracy were almost the same for both settings.



(a) FPR



(b) TPR



(c) Accuracy

Figure 5.4: Comparison between classical and FedAVG setting for the client c_0

5.3 Results in Real-World Scenario

Additionally, we tested our system with real-world data collected on Cohda Boxes in the BeIntelli infrastructure. In the following, we will refer to Cohda Box 2 (with IP: 192.168.230.3) as c_2 and Cohda Box 5 (with IP: 192.168.213.86) as c_5.

Optimal Threshold for Detection Again we measure the FPR in relation to changing values of the detection threshold on normal traffic to determine the optimal threshold for the detection. In Figure 5.5 we see, that the FPR is decreasing rapidly until it starts to converge to 0 when reaching a MAD threshold of 2.2. Therefore, we selected a MAD threshold of 2.2 for the real-world dataset to minimize the FPR without having a decreased anomaly detection performance.

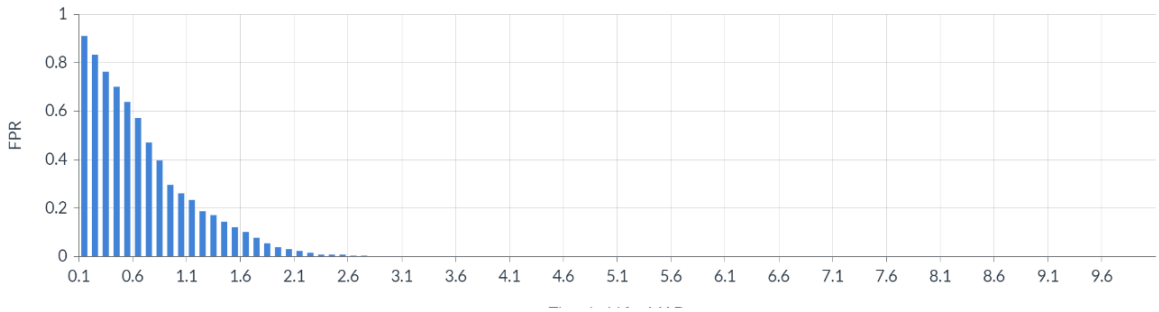


Figure 5.5: FPR based on different MAD thresholds for normal traffic

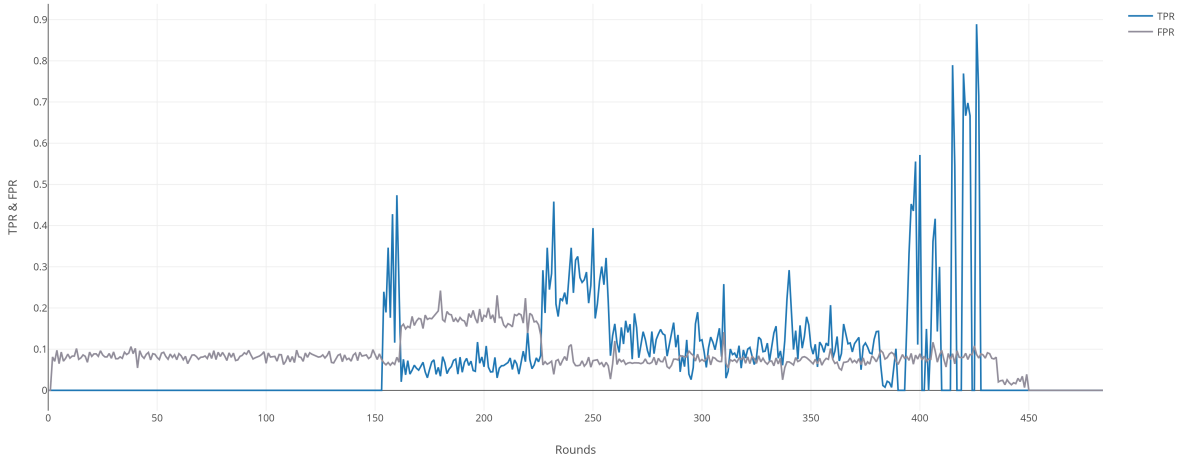
Detection Time & Throughput We measure the minimal, maximal, and average detection time needed between receiving an anomaly and reporting it as malicious (see Table 5.3). On the real-world dataset, the algorithm was a bit slower than on the CIC-IDS17 dataset and this time usually succeeds in detecting the anomalies after around 0.5 seconds. The maximum measured time a client needed for detecting an anomaly was less than 4 seconds which is still low enough to successfully function as an F-IDS in real systems. The best detection time a client reached was close to 0.1 seconds in detecting a Brute Force Response. The first attacks called Installation of Attack Tool and SSH Brute Force were detected in less than 2% and thus surprisingly rare. This could be related to the fact that both attacks are not point anomalies, and therefore difficult to detect by the autoencoder in use. The Installation of Attack Tool where the attacker ultimately only downloaded programs to attack other network devices, cannot be easily detected on the basis of network traffic alone, and needs further contextual information like subsequent malicious activity or detailed tool information. Furthermore, the collective anomaly SSH Brute Force is only detectable in the context of multiple packets, which makes anomaly-based detection with autoencoders difficult. The far more critical attacks in which data has been stolen from a Cohda Box were already detected by the F-IDS in almost 65% of the cases. This demonstrates that our system fulfills its task of quickly detecting attacks that are critical to the system and also proves that there is enough potential to increase the results.

Type of Attack	minimum in seconds	maximum in seconds	average in seconds	Detected	Total	Percentage
Installation Attack Tool	0.1499	2.1508	0.5234	844	56889	1.5%
SSH Brute Force	0.1444	3.3569	0.5427	1008	51090	2.0%
SSH Brute Force Response	0.1018	3.9953	0.5313	13265	49324	26.9%
SSH Privilege Escalation	0.1557	3.5548	0.4663	90	548	16.4%
SSH Data leakage	0.1631	1.4369	0.4908	346	535	64.7%

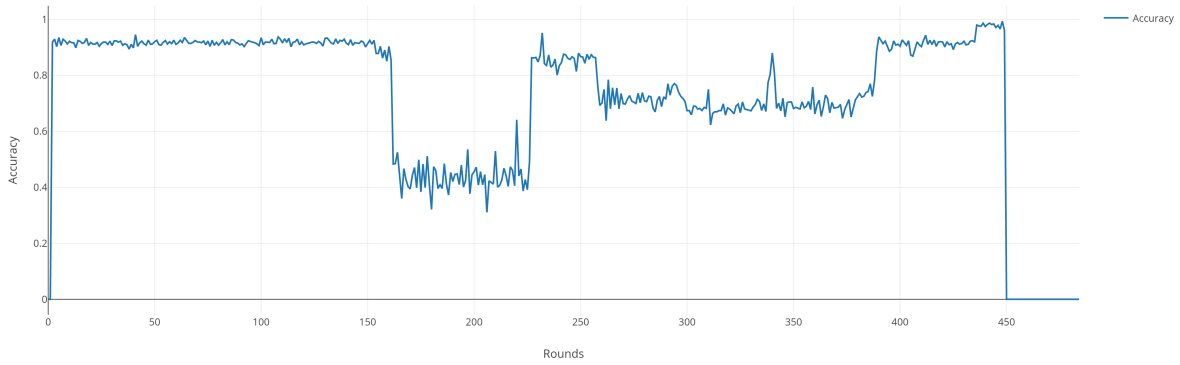
Table 5.3: Time needed by a client between receiving a sample and making a prediction in seconds

Centralized TPR, FPR, Accuracy and F_1-Score Again we calculate the average performance of both clients on the server. Figure 5.6a shows the averaged FPR and TPR values in the real-world setting, Figure 5.6b shows the systems accuracy in each round, and Figure 5.6c the averaged F_1-Score. The FPR is the most time less than 10% and only increases slightly between rounds 160 and 230. Especially in the beginning of normal traffic, the accuracy is close to 90% and reaches even values of 99.3% in the end. Again, we see a significant decrease in accuracy between rounds 160 and 230. We assume that the Installation of Attack Tool anomaly in c_5 has a negative impact on the model performance. It is likely that the system has begun to learn these anomalies and become less accurate at distinguishing normal traffic from anomalies. It could be beneficial to adjust the model parameters like learning rate in later work to prevent the autoencoder from learning the patterns of anomalies too fast [19]. The F_1-Score is getting better over the first 200 rounds and reaches a maximum value of 0.4. This value is relatively low for IDS systems but acceptable since our datasets contain significantly more normal traffic than anomalies and we do not focus on fine-tuning.

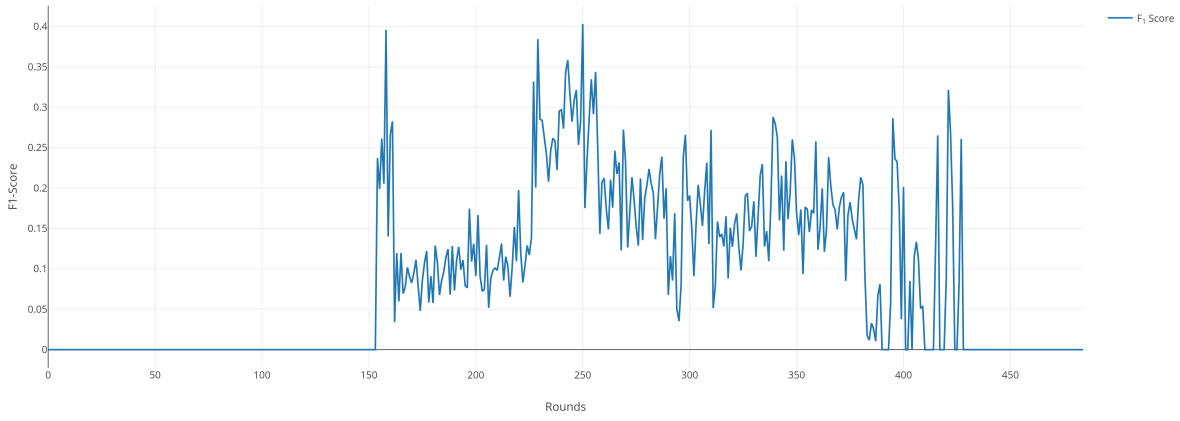
Comparison of Clients Last but not least, we analyzed the performance of the clients c_2 and c_5 separately on the real-world dataset in Figure 5.7. Client c_2 showed a slightly higher FPR than c_5, but it is particularly noticeable that between rounds 160 and 230, the FPR of c_5 increased sharply. Here, a disproportionate amount of normal traffic seems to have been flagged as an anomaly. The TPR of c_5 with a maximum of 0.4 is also significantly worse than that of c_2 with a maximum of 1. Client c_5 achieves a maximum accuracy of 98% measured with normal traffic, the accuracy of c_2 is slightly worse with values of about 85%. In conclusion, it can be said that the developed F-IDS is a good example of the application of FL in intrusion detection for autonomous transportation systems. With the real data collected from the BeIntelli network targeted with simulated attacks, we conducted a detailed analysis of the performance. In some situations with real-world data, the simple autoencoder as a global model is far from sufficient for a good intrusion detection performance. However, our framework offers much potential for improvements to achieve higher accuracy in later developments and showed that an implementation of a F-IDS in autonomous transportation systems is feasible and beneficial.



(a) FPR and TPR

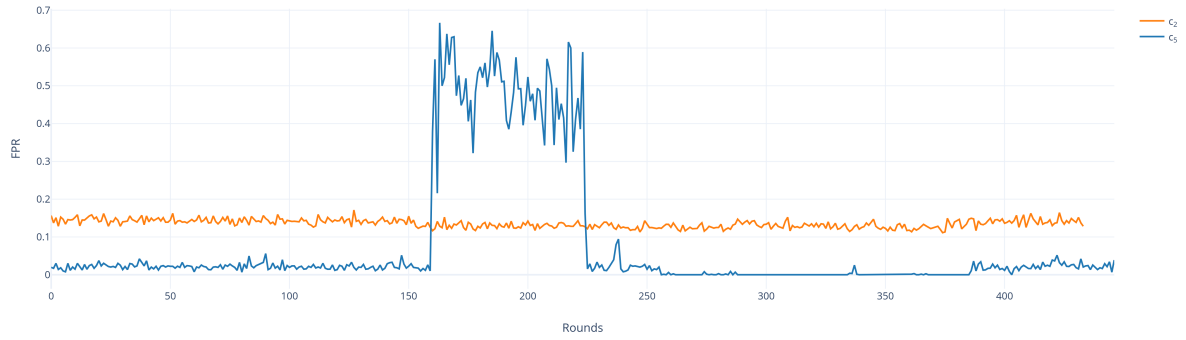


(b) Accuracy

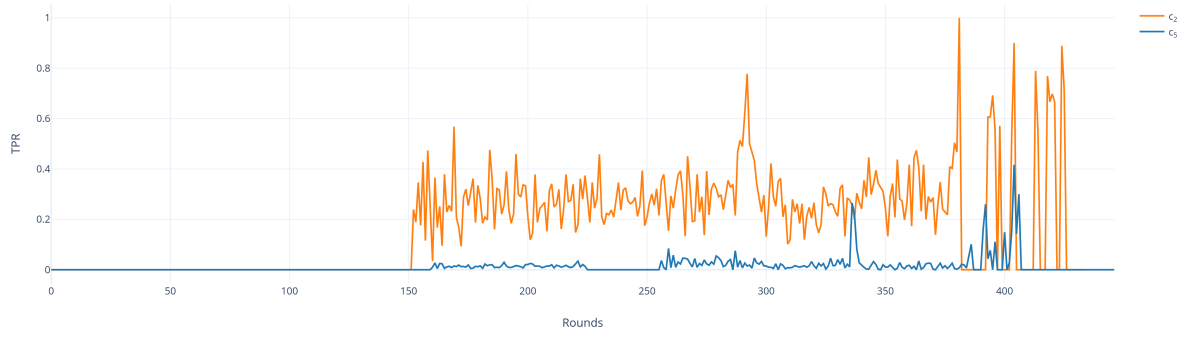


(c) F_1-Score

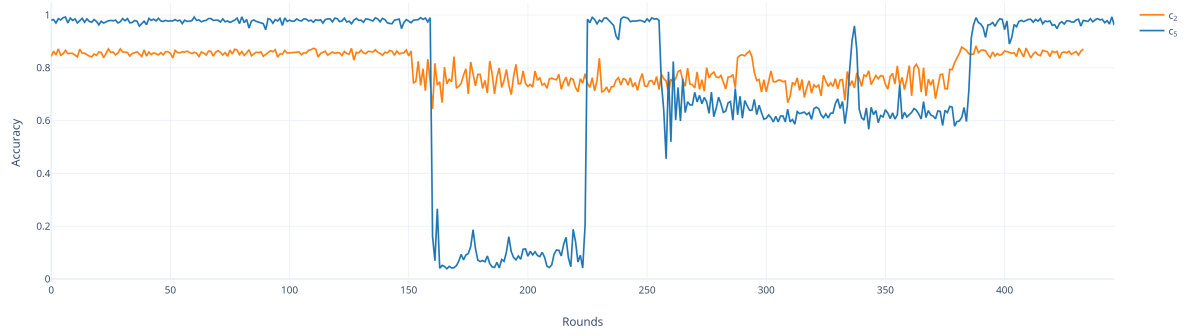
Figure 5.6: Averaged results over both clients c_2 and c_5 in the real-world setting



(a) FPR for client c_2 and c_5



(b) TPR for client c_2 and c_5



(c) Accuracy for client c_2 and c_5

Figure 5.7: Results for the clients based on the real-world dataset

5.4 Discussion

Many modern IDS solutions suffer from limited effectiveness against unknown attacks and insufficient adaptability to different systems (see chapter 3). With our work, we have taken a closer look at some of the problems of FL and IDS and proposed an extensively evaluated solution for federated intrusion detection in autonomous transportation systems. The developed F-IDS uses an autoencoder as a global model, which enables the system to be capable of dealing with more complex situations than most simple pattern based IDS. Many recent approaches of IDS in vehicles focus on the internal CAN bus system vulnerabilities and some systems even achieve a good performance for such a CAN bus systems on old datasets [9]. Unfortunately, many IDS are not improving the pre-trained model over time, which makes the whole system vulnerable to novel attacks or have never been tested on real-world datasets. Some IDS only use a small amount of information from the large available but complex data streams for the detection so that anomalies are searched only on the basis of packet timing [97] or based on measured response times [60]. Other approaches have a limited effectiveness against novel attacks and struggle with the real-time and continuous data stream processing [43]. Our F-IDS tries to solve these problems by providing a high data throughput, a high adaptability to the respective situation, and a good performance, especially with respect to unseen point anomalies. Our system monitors the network data of the distributed components of an autonomous transport system and detects attacks based on pre-selected features that are important for intrusion detection. If required, the data processing could be adapted to the specific areas of application and thus be significantly improved. That FL is applicable in intrusion detection has already been shown by [89] with real-world data in a M-CPS. The proposed F-IDS achieves a detection accuracy of 99.0% and a false positive rate of 1.0% along with a reduced network communication overhead. These results indicate that even autonomous transportation systems will benefit from a F-IDS if the underlying IDS is improved and customized for the situation and deployed. There are various improvement methods and ideas that could be implemented and tested in future developments. Currently, in our centralized setup, the server is the single bottleneck. If this instance fails, the client side IDS will still detect intrusions, but the models are no longer updated by the server. Therefore, the newest developments of FL use a decentralized blockchain approach to mitigate the risk of a single-point failure caused by the central aggregation server [1]. In the future, we could also analyze the performance of other FL aggregation algorithms. A federated meta-learning framework called FedMeta using a parameterized algorithm could provide a reduced communication cost and an increased accuracy with faster convergence [24]. Other researchers were also able to improve the performance results on non-IID data by creating small subsets of data that are globally shared between all the client devices [114]. Similarly, other machine learning methods could be evaluated in our system, to see how they perform in a federated training setting compared to autoencoders. As described in the evaluation, our system suffers from a low detection accuracy for collective anomalies like distributed denial of service (DDoS), why it might be interesting to combine our F-IDS with long-short-term-memory (LSTM) networks that are capable of memorizing the latest data samples. Moreover, it would be recommendable to try various preprocessing steps to improve the model performance. The preprocessing needs to be done for the specific use case and would further increase the model's accuracy. However, it should be considered that in most online learning applications, there is only a limited possibility to apply preprocessing due to limited processing time and unknown data properties.

6
Conclusion

The advancing automation in the transportation sector is accompanied by a big problem: A rising number of cyberattacks all over the world and the highest possibility to be exposed to tremendously harmful and cost-intensive attacks. Researchers and companies are forced to implement countermeasures like intrusion detection systems to automatically detect such existence-threatening cyberattacks as soon as possible. In this thesis, we proposed a new way of intrusion detection in autonomous transportation systems using federated learning. With federated learning, different devices in a network train an improved machine learning model together, enabling autonomous transportation systems to form a fast and continuously learning intrusion detection system. We developed our own implementation of the popular FedAVG algorithm for the aggregation of autoencoder weights trained on multiple clients. We presented the customizable machine learning model, the federated optimization algorithm, and our own communication protocol with encrypted data transmission between the server and the clients. In extensive experiments on the CIC-IDS17 dataset and real-world data mixed with simulated attacks collected in the BeIntelli network, our algorithm achieves a high accuracy of around 90% in normal traffic and a low false positive rate of about 15%. We showed that federated learning allows advancements of up to 20% in true positive rate for individual participating clients over a non-federated training setting. Thus, federated learning for intrusion detection in intelligent cars, trains, buses, and the surrounding infrastructure could provide safety for people traveling around the world in a reliable and secure transportation system. Our framework offers several key advantages for future enhancements and optimizations. The aggregation of the client models can be freely configured and fine-tuned, while our own communication protocol ensures a high reliability and an improved failure recovery. In the future, we will investigate further approaches of federated learning, e.g. decentralized learning, which also eliminates the global server. In addition, the performance of other machine learning models could be evaluated in a federated learning setting together with our real-world network data and precisely simulated attacks in order to significantly improve the results.

Bibliography

[1] Mohamed Abdel-Basset et al. "Federated Intrusion Detection in Blockchain-Based Smart Transportation Systems". In: *IEEE Transactions on Intelligent Transportation Systems* 23.3 (2022), pp. 2523–2537. DOI: 10.1109/TITS.2021.3119968.

[2] Ahmed Abdullah. *Basics of Auto-Encoders Explained*. Aug. 2021. URL: https://medium.com/red-buffer/basics-of-auto-encoders-explained-e0699a3db263.

[3] Adap GmbH. *Evaluation*. Feb. 2023. URL: https://flower.dev/docs/evaluation.html.

[4] Charu C. Aggarwal. *Outlier Analysis*. 2nd. Springer Publishing Company, 2016. ISBN: 3319475770. DOI: https://doi.org/10.1007/978-3-319-47578-3.

[5] Mohiuddin Ahmed. "Collective Anomaly Detection Techniques for Network Traffic Analysis". In: *Annals of Data Science* 5.4 (Dec. 2018), pp. 497–512. ISSN: 2198-5812. DOI: 10.1007/s40745-018-0149-0.

[6] Naveed Akhtar et al. "Advances in Adversarial Attacks and Defenses in Computer Vision: A Survey". In: *IEEE Access* 9 (2021), pp. 155161–155196. DOI: 10.1109/ACCESS.2021.3127960.

[7] Asma Aljohani and Anas Bushnag. "An Intrusion Detection System Model in a Local Area Network using Different Machine Learning Classifiers". In: *2021 11th International Conference on Advanced Computer Information Technologies (ACIT)*. 2021, pp. 483–488. DOI: 10.1109/ACIT52158.2021.9548421.

[8] Rachel Allen and Gorkem Batmaz. *Fingerprinting Every Network User and Asset with NVIDIA Morpheus*. Oct. 2021. URL: https://developer.nvidia.com/blog/fingerprinting-every-network-user-and-asset-with-morpheus/.

[9] Ayoub Alsarhan et al. "Machine Learning-driven optimization for SVM-based intrusion detection system in vehicular ad hoc networks". In: *Journal of Ambient Intelligence and Humanized Computing* (Feb. 2021). ISSN: 1868-5145. DOI: 10.1007/s12652-021-02963-x.

[10] Alstom. *Autonomous transportation: Smart and safe operations*. URL: https://www.alstom.com/solutions/digital-mobility/autonomous-transportation-smart-and-safe-operations.

[11] Simran Mann Andreas Streim. *203 Milliarden Euro Schaden pro Jahr durch Angriffe auf deutsche Unternehmen*. 2022. URL: https://www.bitkom.org/Presse/Presseinformation/Wirtschaftsschutz-2022.

[12] Manoj Ghuhan Arivazhagan et al. "Federated Learning with Personalization Layers". In: *CoRR* (2019). arXiv: 1912.00818. URL: http://arxiv.org/abs/1912.00818.

[13] Anouar Bachar, Noureddine EL Makhif, and Omar EL Bannay. "Towards a behavioral network intrusion detection system based on the SVM model". In: *2020 1st International Conference on Innovative Research in Applied Science, Engineering and Technology (IRASET)*. 2020, pp. 1–7. DOI: 10.1109/IRASET48871.2020.9092094.

[14] Saeed Asadi Bagloee et al. "Autonomous vehicles: challenges, opportunities, and future implications for transportation policies". In: *Journal of Modern Transportation* 24.4 (Dec. 2016), pp. 284–303. ISSN: 2196-0577. DOI: 10.1007/s40534-016-0117-3.

[15] Aitor Belenguer, Javier Navaridas, and Jose A. Pascual. *A review of Federated Learning in Intrusion Detection Systems for IoT*. 2022. DOI: 10.48550/ARXIV.2204.12443.

[16] J. Rene Beulah et al. "Enhancing Detection of R2L Attacks by Multistage Clustering Based Outlier Detection". In: *Wireless Personal Communications* 124.3 (June 2022), pp. 2637–2659. ISSN: 1572-834X. DOI: 10.1007/s11277-022-09482-8.

[17] David Briginshaw. *Italian railway IT system suffers major cyber-attack*. Mar. 2022. URL: https://www.railjournal.com/infrastructure/italian-railway-it-system-suffers-major-cyber-attack/.

[18] Jason Brownlee. *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*. July 2017. URL: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.

[19] Jason Brownlee. *Understand the Impact of Learning Rate on Neural Network Performance*. Sept. 2020. URL: https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/.

[20] Jason Brownlee. *Why One-Hot Encode Data in Machine Learning?* July 2017. URL: https://machinelearningmastery.com/why-one-hot-encode-data-in-machine-learning/.

[21] Bundeskriminalamt. *Cybercrime - Bundeslagebild 2021*. May 2022. URL: https://www.bka.de/SharedDocs/Downloads/DE/Publikationen/JahresberichteUndLagebilder/Cybercrime/cybercrimeBundeslagebild2021.pdf.

[22] Keyan Cao et al. "An Overview on Edge Computing Research". In: *IEEE Access* 8 (2020), pp. 85714–85728. DOI: 10.1109/ACCESS.2020.2991734.

[23] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly Detection: A Survey". In: *ACM Comput. Surv.* 41.3 (July 2009). ISSN: 0360-0300. DOI: 10.1145/1541880.1541882.

[24] Fei Chen et al. "Federated Meta-Learning for Recommendation". In: *CoRR* (2018). arXiv: 1802.07876. URL: http://arxiv.org/abs/1802.07876.

[25] Yiqiang Chen et al. "FedHealth: A Federated Transfer Learning Framework for Wearable Healthcare". In: *CoRR* abs/1907.09173 (2019). arXiv: 1907.09173. URL: http://arxiv.org/abs/1907.09173.

[26] Yujing Chen, Yue Ning, and Huzefa Rangwala. "Asynchronous Online Federated Learning for Edge Devices". In: *CoRR* abs/1911.02134 (2019). arXiv: 1911.02134. URL: http://arxiv.org/abs/1911.02134.

[27] Council of European Union. *Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). Official Journal of the European Union L 119/1, 1–88 (Apr 2016)*. 2016. URL: https://eur-lex.europa.eu/legal-content/DE/TXT/PDF/?uri=CELEX:32016R0679&from=EN.

[28] Antonia Creswell, Kai Arulkumaran, and Anil A. Bharath. "On denoising autoencoders trained to minimise binary cross-entropy". In: *CoRR* abs/1708.08487 (2017). arXiv: 1708.08487. URL: http://arxiv.org/abs/1708.08487.

[29] Joan Daemen and Vincent Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. Jan. 2002. ISBN: 978-3-540-42580-9. DOI: 10.1007/978-3-662-04722-4.

[30] Muataz Salam Al-Daweri et al. "An Analysis of the KDD99 and UNSW-NB15 Datasets for the Intrusion Detection System". en. In: *Symmetry* 12.10 (Oct. 2020). Number: 10 Publisher: Multidisciplinary Digital Publishing Institute, p. 1666. ISSN: 2073-8994. DOI: 10.3390/sym12101666.

[31] Julianna Delua. *Supervised vs. Unsupervised Learning: What's the Difference?* Mar. 2021. URL: https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning.

[32] Arden Dertat. *Applied Deep Learning - Part 3: Autoencoders*. Oct. 2017. URL: https://towardsdatascience.com/applied-deep-learning-part-3-autoencoders-1c083af4d798.

[33] Cynthia Dwork. "Differential Privacy". In: *Automata, Languages and Programming*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 1–12. ISBN: 978-3-540-35908-1. URL: https://link.springer.com/chapter/10.1007/11787006_1.

[34] Ronald Eikenberg and David Wischnjak. *heise Investigativ: Viele Ampeln sind per Funk einfach manipulierbar*. Dec. 2022. URL: https://www.heise.de/hintergrund/heise-Investigativ-Viele-Ampeln-sind-per-Funk-einfach-manipulierbar-7367885.html?seite=all.

[35] Ahmet M. Elbir et al. *Federated Learning in Vehicular Networks*. 2020. DOI: 10.48550/ARXIV.2006.01412.

[36] Heike Flämig. "Autonomous Vehicles and Autonomous Driving in Freight Transport". In: *Autonomous Driving: Technical, Legal and Social Aspects*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2016, pp. 365–385. ISBN: 978-3-662-48847-8. DOI: 10.1007/978-3-662-48847-8_18.

[37] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart. "Model Inversion Attacks That Exploit Confidence Information and Basic Countermeasures". In: *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. CCS '15. Denver, Colorado, USA: Association for Computing Machinery, 2015, pp. 1322–1333. ISBN: 9781450338325. DOI: 10.1145/2810103.2813677.

[38] Bing Gao et al. "An Intrusion Detection Method Based on Machine Learning and State Observer for Train-Ground Communication Systems". In: *IEEE Transactions on Intelligent Transportation Systems* 23.7 (2022), pp. 6608–6620. DOI: 10.1109/TITS.2021.3058553.

[39] Sergiu Gatlan. *Automotive supplier breached by 3 ransomware gangs in 2 weeks*. Aug. 2022. URL: https://www.bleepingcomputer.com/news/security/automotive-supplier-breached-by-3-ransomware-gangs-in-2-weeks/.

[40] Andy Greenberg. *The Jeep Hackers Are Back to Prove Car Hacking Can Get Much Worse*. Aug. 2016. URL: https://www.wired.com/2016/08/jeep-hackers-return-high-speed-steering-acceleration-hacks/.

[41] Kerem Gülen. *Finding loopholes with machine learning techniques*. Oct. 2022. URL: https://dataconomy.com/2022/10/machine-learning-anomaly-detection/.

[42] Sohan Gyawali et al. "Challenges and Solutions for Cellular Based V2X Communications". In: *IEEE Communications Surveys & Tutorials* 23.1 (2021), pp. 222–255. DOI: 10.1109/COMST.2020.3029723.

[43] Roland E. Haas et al. "Intrusion detection in connected cars". In: *2017 IEEE International Conference on Electro Information Technology (EIT)*. 2017, pp. 516–519. DOI: 10.1109/EIT.2017.8053416.

[44] Nuha.A Hamad, Khattab M. Ali Alheeti, and Salah Al-Rawi. "Intrusion detection system using artificial intelligence for internal messages of robotic cars". In: vol. 2400. Oct. 2022, p. 020005. DOI: 10.1063/5.0112275.

[45] Handelsblatt Online. *Das sind die größten Zughersteller der Welt*. Sept. 2017. URL: https://www.wiwo.de/unternehmen/industrie/siemens-alstom-hitachi-und-co-das-sind-die-groessten-zughersteller-der-welt/20392810.html.

[46] David C. Hoaglin and Boris lglewicz. "Volume 16: How to Detect and Handle Outliers". In: ASQ Quality Press, 1993.

[47] IBM. *Average cost of a data breach in the United States from 2006 to 2022*. In Statista. July 2022. URL: https://www.statista.com/statistics/273575/us-average-cost-incurred-by-a-data-breach/.

[48] Muhammad Usman Ilyas, Siddique Latif, and Junaid Qadir. "Using Deep Autoencoders for Facial Expression Recognition". In: *CoRR* abs/1801.08329 (2018). arXiv: 1801.08329. URL: http://arxiv.org/abs/1801.08329.

[49] Matthias Janson. *Viele würden selbstfahrende Verkehrsmittel benutzen*. June 2022. URL: https://de.statista.com/infografik/27564/befragte-die-eines-der-folgenden-autonomen-fahrzeuge-zu-nutzen-wuerden/.

[50] Najeeb Moharram Jebreel et al. *Defending against the Label-flipping Attack in Federated Learning*. July 2022. DOI: 10.48550/ARXIV.2207.01982.

[51] Ban Mohammed Khammas. "Ransomware Detection using Random Forest Technique". In: *ICT Express* 6.4 (2020), pp. 325–331. ISSN: 2405-9595. DOI: https://doi.org/10.1016/j.icte.2020.11.001.

[52] Shah Khalid Khan et al. "Cyber-attacks in the next-generation cars, mitigation techniques, anticipated readiness and future directions". In: *Accident Analysis & Prevention* 148 (2020), p. 105837. ISSN: 0001-4575. DOI: https://doi.org/10.1016/j.aap.2020.105837.

[53] Ansam Khraisat et al. "Survey of intrusion detection systems: techniques, datasets and challenges". In: *Cybersecurity* 2.1 (July 2019), p. 20. ISSN: 2523-3246. DOI: 10.1186/s42400-019-0038-7.

[54] Pavel Klushin. *Online vs offline machine learning – what's the difference?* Jan. 2022. URL: https://www.qwak.com/post/online-vs-offline-machine-learning-whats-the-difference.

[55] Jakub Konečný et al. "Federated Learning: Strategies for Improving Communication Efficiency". In: *CoRR* abs/1610.05492 (2016). arXiv: 1610.05492. URL: http://arxiv.org/abs/1610.05492.

[56] Jakub Konečný et al. “Federated Optimization: Distributed Machine Learning for On-Device Intelligence”. In: *CoRR* abs/1610.02527 (2015). arXiv: 1610.02527. URL: http://arxiv.org/abs/1610.02527.

[57] Ravdeep Kour, Adithya Thaduri, and Ramin Karim. “Railway Defender Kill Chain to Predict and Detect Cyber-Attacks”. In: 9 (Jan. 2020), pp. 47–90. DOI: 10.13052/jcsm2245-1439.912.

[58] M. Kyriakidis, Riender Happee, and Joost de Winter. “Public opinion on automated driving: Results of an international questionnaire among 5000 respondents”. In: *Transportation Research Part F: Traffic Psychology and Behaviour* 32 (July 2015). DOI: 10.1016/j.trf.2015.04.014.

[59] Hyeryun Lee et al. “Fuzzing CAN Packets into Automobiles”. In: *2015 IEEE 29th International Conference on Advanced Information Networking and Applications*. 2015, pp. 817–821. DOI: 10.1109/AINA.2015.274.

[60] Hyunsung Lee, Seong Hoon Jeong, and Huy Kang Kim. “OTIDS: A Novel Intrusion Detection System for In-vehicle Network by Using Remote Frame”. In: *2017 15th Annual Conference on Privacy, Security and Trust (PST)*. 2017, pp. 57–5709. DOI: 10.1109/PST.2017.00017.

[61] Laetitia Leichtnam et al. “Sec2graph: Network Attack Detection Based on Novelty Detection on Graph Structured Data”. en. In: *Detection of Intrusions and Malware, and Vulnerability Assessment*. Lecture Notes in Computer Science. Springer International Publishing, 2020, pp. 238–258. ISBN: 978-3-030-52683-2. DOI: 10.1007/978-3-030-52683-2_12.

[62] Beibei Li et al. “DeepFed: Federated Deep Learning for Intrusion Detection in Industrial Cyber–Physical Systems”. In: *IEEE Transactions on Industrial Informatics* 17.8 (2021), pp. 5615–5624. DOI: 10.1109/TII.2020.3023430.

[63] Yang Liu et al. “Federated Forest”. In: *IEEE Transactions on Big Data* 8.3 (2022), pp. 843–854. DOI: 10.1109/TBDATA.2020.2992755.

[64] Yang Liu et al. “FedVision: An Online Visual Object Detection Platform Powered by Federated Learning”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.08 (Apr. 2020), pp. 13172–13179. DOI: 10.1609/aaai.v34i08.7021.

[65] Yinghui Liu et al. “Blockchain-Enabled Asynchronous Federated Learning in Edge Computing”. In: *Sensors* 21.10 (2021). ISSN: 1424-8220. DOI: 10.3390/s21103335.

[66] Elizabeth MacBride. *The dark web’s criminal minds see Internet of Things as next big hacking prize*. Jan. 2023. URL: https://www.cnbc.com/2023/01/09/the-dark-webs-criminal-minds-see-iot-as-the-next-big-hacking-prize.html.

[67] Brendan McMahan et al. “Communication-Efficient Learning of Deep Networks from Decentralized Data”. In: *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*. ISSN: 2640-3498. PMLR, Apr. 2017, pp. 1273–1282.

[68] Brendan McMahan et al. “Federated Learning of Deep Networks using Model Averaging”. In: *CoRR* abs/1602.05629 (2016). arXiv: 1602.05629. URL: http://arxiv.org/abs/1602.05629.

[69] Anthony Melaragno et al. “Rail Radio Intrusion Detection System (RRIDS) for Communication Based Train Control (CBTC)”. In: *2016 IEEE International Conference on Intelligent Rail Transportation (ICIRT)*. 2016, pp. 39–48. DOI: 10.1109/ICIRT.2016.7588548.

[70] Rob van der Meulen. *What Edge Computing Means for Infrastructure and Operations Leaders*. Oct. 2018. URL: https://www.gartner.com/smarterwithgartner/what-edge-computing-means-for-infrastructure-and-operations-leaders.

[71] Rim Moalla et al. “Risk analysis study of ITS communication architecture”. In: *2012 Third International Conference on The Network of the Future (NOF)*. 2012, pp. 1–5. DOI: 10.1109/NOF.2012.6463997.

[72] Timo Möller et al. *The future of mobility 2020 - McKinsey*. Dec. 2019. URL: https://www.mckinsey.com/industries/automotive-and-assembly/our-insights/the-future-of-mobility-is-at-our-doorstep.

[73] Kostas Mouratidis and Victoria Cobeña Serrano. “Autonomous buses: Intentions to use, passenger experiences, and suggestions for improvement”. In: *Transportation Research Part F: Traffic Psychology and Behaviour* 76 (2021), pp. 321–335. ISSN: 1369-8478. DOI: https://doi.org/10.1016/j.trf.2020.12.007.

[74] Subhojeet Mukherjee et al. “Practical DoS Attacks on Embedded Networks in Commercial Vehicles”. In: vol. 10063. Dec. 2016, pp. 23–42. ISBN: 978-3-319-49805-8. DOI: 10.1007/978-3-319-49806-5_2.

[75] Mohammed Nadeem Hangar et al. “A Survey of Autonomous Vehicles: Enabling Communication Technologies and Challenges”. In: *Sensors* 21 (Jan. 2021), p. 706. DOI: 10.3390/s21030706.

[76] Thien Duc Nguyen et al. “DÏoT: A Crowdsourced Self-learning Approach for Detecting Compromised IoT Devices”. In: *CoRR* abs/1804.07474 (2018). arXiv: 1804.07474. URL: http://arxiv.org/abs/1804.07474.

[77] Breana Noble. *As Cyber Attacks on Cars Rise, So Does Related Cybersecurity*. Sept. 2022. URL: https://www.govtech.com/transportation/as-cyber-attacks-on-cars-rise-so-does-related-cybersecurity.

[78] Iishi Patel. *Facial Reconstruction using Autoencoders*. May 2020. URL: https://towardsdatascience.com/facial-reconstruction-using-autoencoders-ed945def10df.

[79] Le Trieu Phong et al. “Privacy-Preserving Deep Learning via Additively Homomorphic Encryption”. In: *IEEE Transactions on Information Forensics and Security* 13.5 (2018), pp. 1333–1345. DOI: 10.1109/TIFS.2017.2787987.

[80] Kevin Poulsen. *Hacker Disables More Than 100 Cars Remotely*. Mar. 2017. URL: https://www.wired.com/2010/03/hacker-bricks-cars/.

[81] Jinchuan Qian et al. “A review on autoencoder based representation learning for fault detection and diagnosis in industrial processes”. In: *Chemometrics and Intelligent Laboratory Systems* 231 (2022), p. 104711. ISSN: 0169-7439. DOI: https://doi.org/10.1016/j.chemolab.2022.104711.

[82] Reuters. *Danish train standstill on Saturday caused by cyber attack*. Nov. 2022. URL: https://www.reuters.com/technology/danish-train-standstill-saturday-caused-by-cyber-attack-2022-11-03/.

[83] Muhammad Noman Riaz and Adeel Ikram. “Development of a secure SMS application using advanced encryption standard (AES) on android platform”. In: *Int. J. Math. Sci. Comput.(IJMSC)* 4.2 (2018), pp. 34–48.

[84] R L Rivest, L Adleman, and M L Dertouzos. “On Data Banks and Privacy Homomorphisms”. In: *Foundations of Secure Computation, Academia Press* (1978), pp. 169–179.

[85] SAE International. *Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles (SAE J3016)*. Apr. 2021. URL: https://www.sae.org/standards/content/j3016_202104/.

[86] David E. Sanger, Clifford Krauss, and Nicole Perlroth. *Cyberattack Forces a Shutdown of a Top U.S. Pipeline*. May 2021. URL: https://www.nytimes.com/2021/05/08/us/politics/cyberattack-colonial-pipeline.html.

[87] Felix Sattler et al. “Robust and Communication-Efficient Federated Learning From Non-i.i.d. Data”. In: *IEEE Transactions on Neural Networks and Learning Systems* 31.9 (2020), pp. 3400–3413. DOI: 10.1109/TNNLS.2019.2944481.

[88] Sabine Schicketanz. *Cyberangriff aufs Rathaus: Potsdam kappt Internetverbindung der Verwaltung – digitale Bedrohungslage*. Dec. 2022. URL: https://www.tagesspiegel.de/potsdam/landeshauptstadt/cyberangriff-aufs-rathaus-potsdam-kappt-internetverbindung-der-verwaltung--digitale-bedrohungslage-9104707.html.

[89] William Schneble and Geethapriya Thamilarasu. “Attack detection using federated learning in medical cyber-physical systems”. In: *28th International conference on computer communications and networks (icccn)*. 2019, pp. 1–8. URL: https://faculty.washington.edu/geetha/Papers/fedlearningIDS.pdf.

[90] Iman Sharafaldin, Arash Habibi Lashkari, and Ali A. Ghorbani. “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization”. In: *International Conference on Information Systems Security and Privacy*. 2018. URL: https://www.scitepress.org/papers/2018/66398/66398.pdf.

[91] Kamran Siddique et al. “KDD Cup 99 Data Sets: A Perspective on the Role of Data Sets in Network Intrusion Detection Research”. In: *Computer* 52.2 (2019), pp. 41–51. DOI: 10.1109/MC.2018.2888764.

[92] Siemens Mobility GmbH. *Autonomous trams on the lines and in the depot*. URL: https://assets.new.siemens.com/siemens/assets/api/uuid:bc2811c4-3d26-460d-9472-9372d5ce32d7/autonomous-tram.pdf.

[93] Matthew Stewart. *Comprehensive Introduction to Autoencoders*. Apr. 2019. URL: https://towardsdatascience.com/generating-images-with-autoencoders-77fd3a8dd368.

[94] Dimitris Stripelis, Paul M. Thompson, and José Luis Ambite. "Semi-Synchronous Federated Learning for Energy-Efficient Training and Accelerated Convergence in Cross-Silo Settings". In: *ACM Trans. Intell. Syst. Technol.* 13.5 (June 2022). ISSN: 2157-6904. DOI: 10.1145/3524885.

[95] Araz Taeihagh and Hazel Si Min Lim. "Governing autonomous vehicles: emerging responses for safety, liability, privacy, cybersecurity, and industry risks". In: *Transport Reviews* 39.1 (2019), pp. 103–128. DOI: 10.1080/01441647.2018.1494640.

[96] Abdel Aziz Taha and Allan Hanbury. "Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool". In: *BMC Medical Imaging* 15.1 (Aug. 2015), p. 29. ISSN: 1471-2342. DOI: 10.1186/s12880-015-0068-x.

[97] Adrian Taylor, Nathalie Japkowicz, and Sylvain Leblanc. "Frequency-based anomaly detection for the automotive CAN bus". In: Dec. 2015, pp. 45–49. DOI: 10.1109/WCICSS.2015.7420322.

[98] Robin Teuwens. *Anomaly Detection with Auto-Encoders*. 2021. URL: https://www.kaggle.com/code/robinteuwens/anomaly-detection-with-auto-encoders.

[99] Nguyen Manh Thang. "Improving Efficiency of Web Application Firewall to Detect Code Injection Attacks with Random Forest Method and Analysis Attributes HTTP Request". In: *Programming and Computer Software* 46.5 (Sept. 2020), pp. 351–361. ISSN: 1608-3261. DOI: 10.1134/S0361768820050072.

[100] Pu Tian et al. "Towards asynchronous federated learning based threat detection: A DC-Adam approach". In: *Computers & Security* 108 (2021), p. 102344. ISSN: 0167-4048. DOI: https://doi.org/10.1016/j.cose.2021.102344.

[101] Joe Tidy. *Honda's global operations hit by cyber-attack*. June 202. URL: https://www.bbc.com/news/technology-52982427.

[102] Andrew Tomlinson, Jeremy Bryans, and Siraj Shaikh. "Towards Viable Intrusion Detection Methods For The Automotive Controller Area Network". In: Sept. 2018. URL: https://www.researchgate.net/publication/328095520_Towards_Viable_Intrusion_Detection_Methods_For_The_Automotive_Controller_Area_Network.

[103] Silvana Trindade, Luiz F. Bittencourt, and Nelson L. S. da Fonseca. "Management of Resource at the Network Edge for Federated Learning". In: *CoRR* abs/2107.03428 (2021). arXiv: 2107.03428. URL: https://arxiv.org/abs/2107.03428.

[104] G. Vigna et al. "A stateful intrusion detection system for World-Wide Web servers". In: *19th Annual Computer Security Applications Conference, 2003. Proceedings.* 2003, pp. 34–43. DOI: 10.1109/CSAC.2003.1254308.

[105] Vukan R Vuchic. "Urban public transportation systems". In: *University of Pennsylvania, Philadelphia, PA, USA* 5 (2002), pp. 2532–2558. URL: http://www.reconnectingamerica.org/assets/Uploads/20020114urbanpubtrsysVuchic.pdf.

[106] Hongyi Wang et al. "Federated Learning with Matched Averaging". In: *CoRR* abs/2002.06440 (2020). arXiv: 2002.06440. URL: https://arxiv.org/abs/2002.06440.

[107] Xiaojie Wang et al. "Deep Learning-Based Network Traffic Prediction for Secure Backbone Networks in Internet of Vehicles". In: *ACM Trans. Internet Technol.* 22.4 (Nov. 2022). ISSN: 1533-5399. DOI: 10.1145/3433548.

[108] Chenhao Xu et al. "Asynchronous Federated Learning on Heterogeneous Devices: A Survey". In: *CoRR* abs/2109.04269 (2021). arXiv: 2109.04269. URL: https://arxiv.org/abs/2109.04269.

[109] Dongliang Xuan et al. "Intrusion Detection System Based on RF-SVM Model Optimized with Feature Selection". In: *2021 International Conference on Communications, Computing, Cybersecurity, and Informatics (CCCI)*. 2021, pp. 1–5. DOI: 10.1109/CCCI52664.2021.9583206.

[110] Qiang Yang et al. "Federated Machine Learning: Concept and Applications". In: *ACM Trans. Intell. Syst. Technol.* 10.2 (Jan. 2019). ISSN: 2157-6904. DOI: 10.1145/3298981.

[111] Xue Ying. "An Overview of Overfitting and its Solutions". In: *Journal of Physics: Conference Series* 1168.2 (Feb. 2019), p. 022022. DOI: 10.1088/1742-6596/1168/2/022022.

[112] Jiong Zhang and Mohammad Zulkernine. "Anomaly Based Network Intrusion Detection with Unsupervised Outlier Detection". In: *2006 IEEE International Conference on Communications*. Vol. 5. 2006, pp. 2388–2393. DOI: 10.1109/ICC.2006.255127.

[113] Lu Zhang et al. "A real-time intrusion detection system based on OC-SVM for containerized applications". In: *2021 IEEE 24th International Conference on Computational Science and Engineering (CSE)*. 2021, pp. 138–145. DOI: 10.1109/CSE53436.2021.00029.

[114] Yue Zhao et al. “Federated Learning with Non-IID Data”. In: *CoRR* abs/1806.00582 (2018). arXiv: 1806.00582. URL: http://arxiv.org/abs/1806.00582.