




Hausarbeit

Comparison of an established supervised and an unsupervised machine learning model with a novel self-supervised framework for network anomaly detection using the CIC-IDS2017 dataset

vorgelegt am 26. August 2025
an der
Hochschule für Wirtschaft und Recht Berlin

Von:	Cedric Schuster
Matrikelnummer:	77206368628
Fachrichtung:	Wirtschaftsinformatik
Studienjahrgang:	WI23A
Semester:	4. Semester
Ausbildungsbetrieb:	Deloitte Wirtschaftsprüfungsgesellschaft GmbH
Betreuender Prüfer:	Prof. Dr.-Ing. Sebastian Schlesinger
Betreuer Betrieb:	Arne Möller
Unterschrift Betreuer:	

Abstract

This paper investigates the application of machine learning for anomaly-based intrusion detection systems using the CIC-IDS2017 dataset. Three representative approaches are implemented and compared: a supervised Random Forest, an unsupervised Isolation Forest, and the recent Encrypted Traffic Self-Supervised Learning framework (ET-SSL). The experiment evaluates each model under a standard closed-set scenario and a zero-day setting, where selected attack classes are withheld during training. Results show that the Random Forest achieves near-perfect discrimination with macro F1-scores above 0.99 and minimal false-alarm rates, confirming and surpassing related work through optimized preprocessing and threshold tuning. ET-SSL demonstrates moderate but robust performance, generalizing to unseen attacks better than the Isolation Forest, though it remains below state-of-the-art deep autoencoder frameworks. The Isolation Forest consistently underperforms, missing more than half of malicious traffic due to violated assumptions about anomaly rarity. These findings suggest that supervised ensemble methods remain highly effective for datasets such as CIC-IDS2017, while self-supervised frameworks like ET-SSL offer promising avenues for adaptive detection in evolving network environments.

Contents

Code Availability	V
Glossary	VI
List of Figures	VIII
List of Tables	IX
1 Introduction	1
2 Theoretical Background	3
2.1 Networks, Threats and Intrusion Detection Systems	3
2.2 Anomalies in Network Traffic	5
2.3 Anomaly Detection by Machine Learning Classification	6
2.4 CIC-IDS2017 Dataset	10
3 Related Work	12
4 Experiment	16
4.1 Assumptions and Notes	16
4.2 Selected Algorithms	17
4.2.1 Supervised Random Forest	17
4.2.2 Unsupervised Isolation Forest	18
4.2.3 Encrypted Traffic Anomaly Detection using Self-supervised Contrastive Learning	20
4.3 Data Preprocessing	21
4.4 Validation and Metrics	22
5 Results	24
5.1 Exploratory Dataset Analysis	24
5.2 Findings	25
5.3 Closed-Set Results (Base Scenario)	25
5.4 Open-Set Results (Zero-Day Scenario)	26
5.5 Discussion	27

6 Conclusion	29
6.1 Summary	29
6.2 Answer to the Research Questions	29
6.3 Future Research	30
Bibliography	31
A Appendix	34
Declaration	38

Code Availability

The complete source code, including all data preprocessing, model training, and evaluation scripts, is available in the corresponding GitHub repository. Due to length and readability constraints, the full code is not included in this thesis.

Repository Link:

<https://github.com/ProfessorSchuster/Hausarbeit-Paper-Comparison-3-Algorithms-NIDS-CICIDS2017.git>

Glossary

AE Autoencoder

A-NIDS Anomaly-based Network Intrusion Detection Systems

AUC Area Under the Curve

CICIDS CIC-CDS2017

DDos Distributed Denial of Service

DT Decision Tree

ET-SSL Encrypted Traffic Anomaly Detection using Self-supervised Contrastive Learning

FNR False Negative Rate

FPR False Positive Rate

GMM Gaussian Mixture Model

IF Isolation Forest

LR Logistic Regression

ML Machine Learning

NB Naive Bayes

NIDS Network Intrusion Detection Systems

NN Neural Networks

OSI ISO Open Systems Interconnection Reference Model

PCA Principal Component Analysis

RF Random Forest

ROC Receiver Operating Characteristic

SHAP Shapley Additive Explanation

SMOTE Synthetic Minority Over-sampling Technique

SSL Self-Supervised Learning

SVM Support Vector Machines

TNR True Negative Rate

TPR True Positive Rate

xAI Explainable Artificial Intelligence

List of Figures

1	Simplified ISO/OSI Model	4
2	Confusion Matrix	8
3	Decision Threshold	9
4	Receiver Operating Characteristic (ROC) Curve and selected Threshold	9
5	Precision-Recall Curve and selected Threshold	10
6	Conceptual illustration of a Random Forest	18
7	Conceptual illustration of an Isolation Forest	19
8	Contrastive learning process and Embedding Space creation	21
9	Confusion matrices (base scenario): RF (left), IF (center), ET-SSL (right).	26
10	ROC (left) and PR (right) for the base scenario.	26
11	Confusion matrices (zero-day scenario): RF (left), IF (center), ET-SSL (right).	27
12	ROC (left) and PR (right) for the zero-day scenario.	27
13	Macro-F1 overview by approach and scenario.	28

List of Tables

1	Related Work	15
2	CIC-IDS2017 Class Distribution	24
3	All CIC-IDS2017 Features	36
4	All Findings	37

1 Introduction

The use of the internet has become widespread within the last 20 years. In 2024, five billion people accessed the internet, which represents 68% of world's population according to International Telecommunication Union (2024, p. 1). Global broadband traffic exceeded seven zettabytes in 2024 (International Telecommunication Union 2024, p. 27) (1 zettabyte = 1 trillion gigabytes) underlining the importance of a reliable and secure network infrastructure to effectively support the growing digital landscape and protect against any threats. It is often assumed that the existing security measures are sufficient to address these challenges. However, cyberattacks not only remain prevalent and are becoming increasingly sophisticated. The financial damage caused by data breaches rose by 29% to 266 billion euros in Germany from 2023 to 2024 (ENISA 2024, p. 72). According to the ENISA Threat Landscape 2024, attacks have become more frequent and complex: The number of Distributed Denial of Service (DDoS) attacks has increased by 50% within the last year, with peak attack volume doubling to 1.5 terabits per second from 2020 to 2023 (ENISA 2024, p. 80–81). Therefore, there is a need for continuous adaptation and enhancement of security measures, as well as the development of new strategies to mitigate these risks.

This paper investigates Machine Learning (ML)-based Network Intrusion Detection Systems (NIDS) as an approach to improve network security. As there is an ongoing development of new strategies, the overall goal is to compare two established ML-algorithms to a novel framework called Encrypted Traffic Anomaly Detection using Self-supervised Contrastive Learning (ET-SSL) (Sattar et al. 2025) using the publicly available CIC-CDS2017 (CICIDS) dataset (Sharafaldin, Habibi Lashkari, and Ghorbani 2018). One objective is to assess the improvements made with regard to effectiveness, efficiency, and practical applicability in real-world contexts. Therefore, several metrics such as the Receiver Operating Characteristic (ROC), the False Positive Rate (FPR) as well as characteristics such as the adaptability to new threats will be evaluated. The ML-experiment involves the implementation of a supervised, an unsupervised, and a self-supervised network anomaly classifier.

The research is driven by the following major question:

RQ1: How do an established supervised, an unsupervised, and a novel self-supervised machine learning approach perform in network anomaly detection on the CICIDS dataset?

Furthermore, this paper aims to answer these sub-questions:

- RQ2: How do the approaches compare across detection metrics such as ROC-Area Under the Curve (AUC), accuracy, precision, recall, and F1-score?
- RQ3: How do training and inference times vary between the approaches?
- RQ4: How well do the approaches detect previously unseen (zero-day) attack types within the dataset?
- RQ5: What implications do the approaches have for continuous training and operational deployment in practice?

2 Theoretical Background

This chapter provides the theoretical foundation for the following experiment. It covers the concepts of networks and threats, the idea of anomalies in network traffic and how to detect them using ML-enabled Anomaly-based Network Intrusion Detection Systems (A-NIDS). Different types of ML-algorithms are presented, including supervised, unsupervised and self-supervised and how they can be evaluated. The chapter concludes by explaining the underlying concept of the CICIDS dataset. The majority of the presented concepts will not be explained in detail as this would go beyond the scope of this work.

2.1 Networks, Threats and Intrusion Detection Systems

There are many approaches to defining a network. This paper adopts Tanenbaum's definition of a computer network as "a collection of autonomous computers interconnected by a single technology" (Tanenbaum 2011, p. 2), where a computer can be almost any electrical device capable of processing and communicating data (e.g. smartphones) (Tanenbaum 2011, p. 10). The communication of one or multiple computers at a certain point in time or within a timeframe can be defined as network traffic (Buck 2020, p. 19). In the following, the exchange of information will be referred to in this paper as traffic that may be either benign or malicious. To enable a seamless traffic flow, networks rely on standardized rules called protocols. As having a single protocol per network is usually impractical, there are multiple in use, each serving a specific purpose. The different protocols are organized in layers, one of the most well-known being the ISO Open Systems Interconnection Reference Model (OSI) which consists of seven layers (Tanenbaum 2011, p. 29–45). Figure 1 presents the most important layers for this paper. At the application layer the actual content, namely payload, is added to a single element of traffic using e.g. the Simple Mail Transfer Protocol for a mail exchange. Today, payloads are increasingly encrypted and can only be read by the sending and the receiving computer (Sattar et al. 2025, p. 1). All subsequent layers add their information (based on the chosen protocol) to the payload as a header. At the transport layer, the information on how to deliver the payload is added with the most-used protocols being UDP and TCP which e.g. set the error handling and potential retransmission for the current connection. The network layer is responsible for the routing of the traffic and therefore adds information like source and destination IP addresses resulting in a packet. At the data link layer, the packet is encapsulated in a frame which contains information for the actual transmission over the physical medium (e.g. Ethernet or Wi-Fi) like

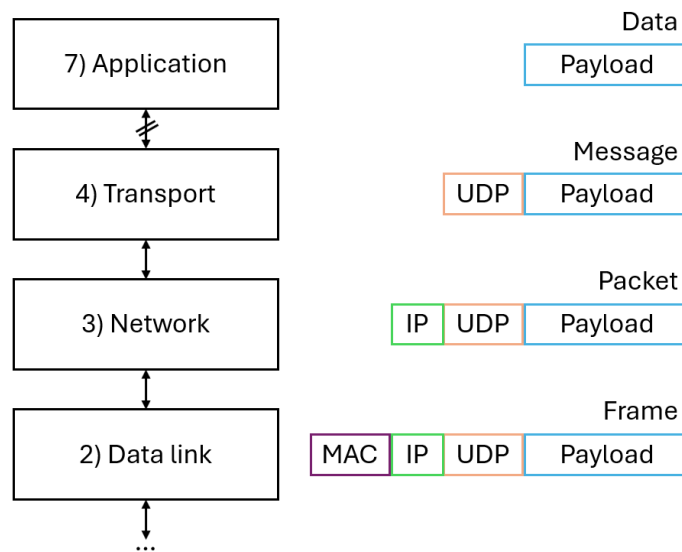


Figure 1: Simplified ISO/OSI Model

the MAC-addresses (Tanenbaum 2011, p. 29–45). All the headers added to the payload contain information that may be useful to determine if the traffic is benign or malicious. Therefore an access point to the network is crucial where the traffic is monitored. This is typically done at gateways (Tanenbaum 2011, p. 28). A potential network topology may be the star topology which allows for a centralized control and thus provides an additional security measure (Buck 2020, p. 18). This will be important later, when explaining where the data for the network traffic analysis comes from.

There are multiple motivations for attacking networks or the traffic flowing through them. Nowadays, the prime threats are ransomware, malware, social engineering, data breaches, DDos attacks and information manipulation (ENISA 2024, p. 8). There are diverse ways to perform such an attack e.g. via accessing or connecting to a network which leads to malicious network traffic. Software systems which try to detect such malicious traffic are called NIDS (Kohli and Chhabra 2025, p. 6). Generally, there are two different NIDS-architectures: Host-NIDS and Network-NIDS. A NIDS implemented at the host level usually performs better as it can analyze the traffic more closely and take into account the specific context of the host or files and logs available on the host computer (Kohli and Chhabra 2025, p. 6–8). The major disadvantages are the high cost of resources as the host-NIDS runs on every computer and the fact that the malicious traffic has already affected the host in some way. Therefore, no real prevention is possible (Buck 2020, p. 22–24). A network-NIDS on the other hand operates at, e.g., a gateway where it can monitor the entire network traffic and therefore has the potential to detect attacks that may not be visible from a single host perspective (Kohli and Chhabra 2025, p. 8). Additionally, it can prevent malicious traffic from accessing the host computer. There are two approaches in which NIDS try to detect malicious traffic. A signature-based approach

compares the total traffic against a known set of malicious signatures (Kohli and Chhabra 2025, p. 8). Even though it is great in the detection of known threats and has a low false detection rate, unknown (zero-day) attacks cannot be detected as the database usually does not contain the newest attack's signature (Buck 2020, p. 22–24). The second concept will be explained in the following Section 2.2.

2.2 Anomalies in Network Traffic

Anomaly is a widely-used term for the description of abnormal or unexpected behavior or simplified, something that is different from the usual (Cambridge Dictionary 2025). In the context of network security, the traffic over a certain time period could be collected and stored as a dataset. In regard to the given definition, this dataset will most likely contain anomalies. One scientific definition of anomalies in a dataset consists of two characteristics. The first is that “anomalies are different from the norm with respect to their features and they are rare in a dataset compared to normal instances” (Goldstein and Uchida 2016, p. 2). As stated in Section 2.1, there are two approaches for NIDS with one being signature comparison. The other one is the detection of anomalies. Three assumptions need to be made to use anomaly detection for NIDS: Malicious traffic is rare, malicious traffic is anomalous and anomalies are assumed to be malicious (Gates and Taylor 2007, p. 3). With respect to the ML-A-NIDS implemented later, the terms instance and feature will be used. An instance is a single data point in the dataset or one single element of network traffic, while a feature is a specific attribute or characteristic of that data point. In the context of network traffic, features are set by the protocols in use and can be differentiated by the layer at which they are added to the payload (Géron 2017, p. 20–26). There are four ways in which anomalies can appear (note: a group of instances is called a cluster) (Buck 2020, p. 5–6):

- **Point or global anomalies:** A single instance is anomalous compared to the rest of the dataset. It can be detected by separating the instance with respect to its features in a multi-dimensional feature space.
- **Local anomaly:** A single instance that is normal in the global context but anomalous in regard to the cluster it belongs to.
- **Collective anomaly:** A group of instances or a cluster where each instance seems normal as it belongs to a cluster. The cluster itself is considered anomalous in the global context as it differs from the major cluster.
- **Contextual anomalies:** Instances which can only be understood as anomalous in a specific context, making them appear normal without further knowledge.

An A-NIDS usually consists of four components, the decoder to gather network traffic data, the preprocessor, the decision engine and the alert system. The data preprocessing pipeline includes e.g. outlier removal but also complex techniques like feature selection or dimensionality reduction. The decision engine classifies instances based on the chosen approach and thus can trigger the alert system which escalates the incident to further algorithms or administrators (Buck 2020, p. 26). To summarize this section, malicious network traffic can be detected using A-NIDS when assuming benign traffic is normal and malicious traffic constitutes anomalies.

2.3 Anomaly Detection by Machine Learning Classification

As stated in Chapter 1, global broadband traffic exceeded seven zettabytes in 2024 (International Telecommunication Union 2024, p. 27). Therefore network classification and malicious traffic detection cannot be performed by human labour. Software solutions are required to implement these approaches. There are statistical techniques (using e.g. mean, median, standard deviation) or expert systems which rely on highly complex pre-defined rules to detect anomalies (Kohli and Chhabra 2025, p. 8). The third approach involves ML (Kohli and Chhabra 2025, p. 8), which is defined as the “science of programming computers so they can learn from data” (Géron 2017, p. 20). Instead of humans finding rules which then lead to a successful A-NIDS, in ML the system fulfills this specific task by having a certain experience. Therefore, a ML-algorithm always needs training data (Géron 2017, p. 20). The performed task is called classification which means that the algorithm assigns a label to an instance or more formal the algorithm predicts the class to which the instance belongs (Géron 2017, p. 114). There can be binary labels like “malicious” or “benign” or multi-class labels such as “DDos”, “PortScan” and “Benign” (Géron 2017, p. 134). There are many different ways in which ML-classifiers can be designed and trained.

- **Supervised learning:** These algorithms are trained on labeled data which means each instance is associated with the desired solution (Géron 2017, p. 26). The algorithm learns by finding patterns in the training data in regard to many examples of the existing classes. The most common algorithms are Decision Tree (DT), Random Forest (RF), Support Vector Machines (SVM), Neural Networks (NN) and regression algorithms (Géron 2017, p. 20–21).
- **Unsupervised learning:** These algorithms are trained on unlabeled data, meaning the system tries to learn the underlying structure of the data without any explicit instructions on what to predict. This can be done by mapping the instances in a high-dimensional feature space. Common techniques include clustering (e.g. k-Means), dimensionality reduction (e.g. Principal Component Analysis (PCA)) or Isolation Forest (IF) (F. T. Liu, Ting, and Z.-H. Zhou 2012, p. 1–3). This is especially useful if the structure is unknown,

high-dimensional or to gain new insights. Additionally, there are high costs related to labelling the data which makes unsupervised learning more attractive. It should be noted that the algorithm testing and verification procedure still needs labeled data (Buck 2020, p. 10). Anomaly detection is a common application of unsupervised learning because the algorithm learns what is considered normal and classifies all other instances as anomalous (Géron 2017, p. 27–30). In regard to the dimensional reduction algorithms, it should be mentioned that these techniques are often applied during data pre-processing, which is explained later in detail (Abdulhammed et al. 2019, p. 1).

- **Semi-supervised learning:** Algorithms that learn from a small number of labeled instances and a large number of unlabeled instances are referred to as semi-supervised. E.g. the labeled part may be normal instances, providing the normal state to the algorithm, which is then refined as the algorithm learns from additional unlabeled data (Buck 2020, p. 10). A different approach is deep belief networks which are based on many unsupervised algorithms (restricted Boltzmann machines) which are stacked on top of each other. Their entirety gets fine-tuned using labeled data (Géron 2017, p. 31).
- **Self-Supervised Learning (SSL):** In general, SSL is a ML-paradigm where the system learns its own supervisory signals from unlabeled input data. Compared to traditional unsupervised learning where algorithms analyze the raw feature space to find patterns or clusters, the SSL-approach aims to learn task-driven representations from the data. This results in more discriminative embeddings and an optimized feature space, called the embedding space. It is achieved by creating pretext tasks, where the model learns to predict part of the input from other parts (Sattar et al. 2025, p. 3). Autoencoder (AE) are able to fulfill this task by reconstructing the input data. After creating an embedding space containing the representations, approaches such as distance measures or IF can be applied to detect anomalies in the embedding space.
- **Contrastive learning:** This is a specific type of SSL where the model learns to differentiate between similar and dissimilar instances. The optimization of the raw feature space is achieved by contrasting positive and negative pairs. A positive pair consists of the original instance and a transformed version of it. E.g. by masking some features or adding noise, an augmentation is created that preserves the semantic meaning of the instance. Positive pairs are clustered closely together in the embedding space, negative pairs are further away which enables a better separation of normal and anomalous.

Regardless of the chosen approach, the performance of every ML-algorithm needs to be quantified for comparison. In the context of A-NIDS, there are two types of error which lead to an ineffective network security solution. First, incorrectly detecting benign traffic as malicious causing many rejections or a large amount of manual work. Secondly, incorrectly detecting malicious traffic as benign resulting in an unsafe and therefore ineffective NIDS. To visualize

		Predicted	
		Negative	Positive
Actual	Negative	TN	FP
	Positive	FN	TP

Figure 2: Confusion Matrix

an algorithm's classification performance, there is the confusion matrix shown in Figure 2 which compares the predicted labels with the true labels (Géron 2017, p. 123). Even though the classifier could perform a multi-class classification task, the decision can always be interpreted as binary because an instance is either benign or malicious. Therefore there are four possible outcomes (Géron 2017, p. 122):

- **True Positive (TP):** The instance is malicious and correctly classified as malicious.
- **True Negative (TN):** The instance is benign and correctly classified as benign.
- **False Positive (FP):** The instance is benign but incorrectly classified as malicious. Also called type I error.
- **False Negative (FN):** The instance is malicious but incorrectly classified as benign. Also called type II error.

As the confusion matrix usually presents the total numbers, each cell can be divided by the total number of positives or negatives:

- **True Positive Rate (TPR):** $\frac{TP}{P} = \frac{TP}{TP+FN} = 1 - FNR$. The proportion of actual positives that are correctly identified (sensitivity/recall/detection rate).
- **True Negative Rate (TNR):** $\frac{TN}{N} = \frac{TN}{TN+FP} = 1 - FPR$. The proportion of actual negatives that are correctly identified (specificity).
- **False Positive Rate (FPR):** $\frac{FP}{N} = \frac{FP}{FP+TN} = 1 - TNR$. The proportion of actual negatives incorrectly identified as positives (type I error rate).
- **False Negative Rate (FNR):** $\frac{FN}{P} = \frac{FN}{FN+TP} = 1 - TPR$. The proportion of actual positives incorrectly identified as negatives (type II error rate).

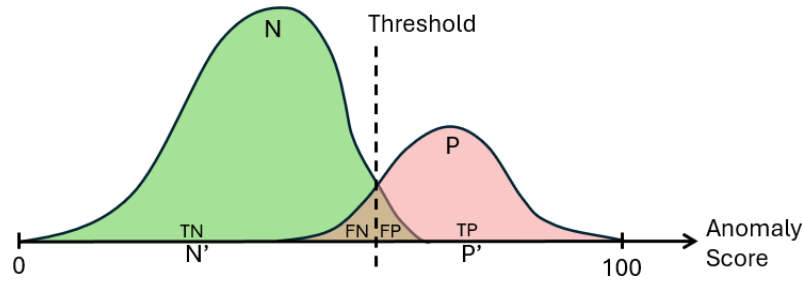


Figure 3: Decision Threshold

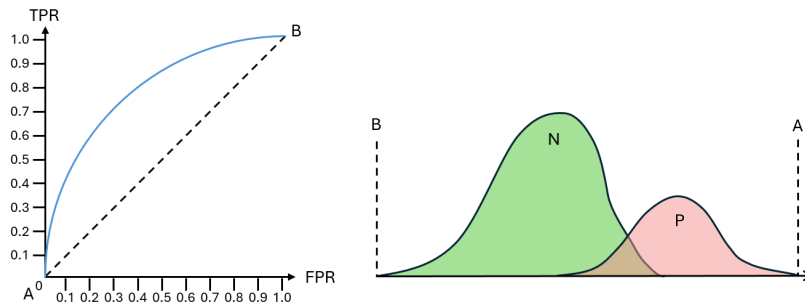


Figure 4: Receiver Operating Characteristic (ROC) Curve and selected Threshold

Another metric to assess the overall correctness of a model is the accuracy, defined as $ACC = \frac{TP+TN}{P+N}$ (Géron 2017, p. 121). The proportion of true results (both true positives and true negatives) among the total number of cases examined. In general, there is no perfect accuracy, in most realistic scenarios, a near 100% accuracy suggests a high chance of over-fitting (Géron 2017, p. 50). There will always be some false positives and some false negatives but to decide if the algorithm should decide towards a better True Negative Rate (TNR) or True Positive Rate (TPR) a threshold is needed as shown in Figure 3. The algorithm provides an anomaly score (a probability that the instance is anomalous) and the threshold determines the classification (Géron 2017, p. 126). As the threshold position may differ depending on the algorithm's use-case, there are approaches to evaluate its performance for all possible thresholds. The ROC-curve is a graphical representation of the true positive rate against the false positive rate for all possible thresholds. The AUC is a single value summarizing the performance of the algorithm, where 1 means perfect classification and 0.5 means random guessing (Géron 2017, p. 130–131). Therefore, the ROC-curve is always above the diagonal line and ideally approaches the point (0,1) as shown in Figure 4. Regarding the edge cases, setting it to point A leads to no anomalies classified, meaning both FPR and TPR are both 0. Setting it to point B means everything is classified as anomalous, resulting in a FPR of 1 and a TPR of 1. The ROC has a major disadvantage for imbalanced datasets because the FPR can remain relatively low due to the high number of negative instances (Buck 2020, p. 17–18). By choosing a low threshold (towards point B), the TPR gets higher but the FPR may still appear small. The precision-recall curve faces this challenge because it only considers the positive class. It plots the precision

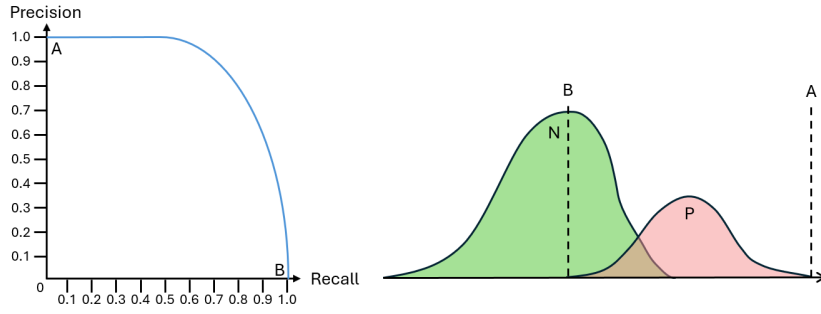


Figure 5: Precision-Recall Curve and selected Threshold

$= \frac{TP}{TP+FP}$ (positive predictive value or correct alarm rate) against the recall (TPR) for different thresholds. As shown in Figure 5, for a high threshold towards point A, the precision is high but the recall is low, while for a low threshold towards point B, the recall is high but the precision is low (Géron 2017, p. 126–129). An optimal precision-recall curve approaches the point (1,1), with the AUC approaching 1. Lastly, there is the F1 score, the harmonic mean of precision and recall, which is defined as $F_1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$ (Géron 2017, p. 125). If the dataset has multiple classes, all presented metrics can be assessed in two ways by either considering the class balance or ignoring it. This means that either the mean will be used to calculate the overall metric for an algorithm (macro metric) or the weighted mean is used, incorporating the number of instances per class as weights.

2.4 CIC-IDS2017 Dataset

The CICIDS dataset is a publicly available dataset for network traffic analysis and anomaly detection. It was created by the Canadian Institute for Cybersecurity at the University of New Brunswick in July 2017 and contains a wide range of network traffic scenarios, including normal traffic, various types of attacks, and background noise (Canadian Institute for Cybersecurity 2018). The dataset was created in connection with the paper by Sharafaldin, Habibi Lashkari, and Ghorbani (2018) following the criteria for a valid NIDS-dataset from Gharib et al. (2016, p. 1–6). Namely, it provides attack diversity, anonymity, available protocols, complete capture, complete interaction, complete network configuration, complete traffic, feature set, heterogeneity labelling, and metadata. The data was gathered in a hybrid manner, meaning network traffic packets were grouped on a flow level and additionally enriched with the payload (Oyelakin et al. 2023, p. 4). Thus, the dataset provides statistical features and does not rely solely on payload data, which is no longer state-of-the-art due to the rise of end-to-end encrypted traffic (Sattar et al. 2025, p. 1). The payloads are available in PCAP format and the overall dataset as CSV files. The dataset’s testbed consists of an attack-network and a victim-network to provide a benign profile (using e.g. HTTP, FTP, SSH protocols) and several attack scenarios (Sharafaldin, Habibi Lashkari, and Ghorbani 2018, p. 1). The overall significance of the CICIDS dataset

is underlined by more than 5,000 citations of the paper by Sharafaldin, Habibi Lashkari, and Ghorbani (2018) as of August 2025. It is still representative and well-suited for the evaluation of NIDS to this day (Oyelakin et al. 2023, p. 4). Nonetheless, Engelen, Rimmer, and Joosen (2021) critically assessed the dataset's limitations and found several issues such as the misimplementation of some attack types, a lack of documentation, mislabelled instances and errors in the feature extraction procedure (Engelen, Rimmer, and Joosen 2021, p. 7). Further dataset details are provided in Chapter 4 when conducting an exploratory analysis.

3 Related Work

The related work in the field of ML-A-NIDS is broad and heterogeneous, with numerous approaches and methodologies being proposed. This chapter reviews approaches validated on the CICIDS dataset and therefore allow for a comparison. Their approaches and results can be seen in Table 1. The approaches demonstrate varying levels of performance, with recently proposed or more advanced methods achieving higher F1-Scores and ROC-AUC values. Nonetheless, the class imbalance in CICIDS seems to be an issue for many algorithms as some attacks remained undetected in certain studies. Data preprocessing, feature selection or dimensionality reduction show overall improvements.

Study	Preprocessing	ML algorithm(s)	Best results	Further results
Al Lail, Garcia, and Olivo (2023): Comparative NIDS study	Cleaning; label conversion; Synthetic Minority Over-sampling Technique (SMOTE); MinMaxScaler; random & stratified split	RF; SVM; Logistic Regression (LR); Gaussian Naive Bayes (NB)	RF: macro Precision/Recall/F1-Score = 0.97	Other results: macro = 0.14–0.25; weighted = 0.69–0.90. Results indicate poor performance on imbalanced data.
Alotaibi and Mafeis (2024): Deep-AE framework	Not specified	Offline deep-AE framework	F1-Score = 92.22; Accuracy = 89.69; AUC-ROC = 96.79	Not specified
Carrera et al. (2022): Unsupervised/SSL-approach with real-time evaluation	Not specified	Deep AE combined with: (1) Gaussian Mixture Model (GMM) & IF; (2) Memory Augmented Deep AE with IF	Memory Augmented Deep AE: F1-Score = 0.97; ROC-AUC = 0.7146; Accuracy = 0.8194	Not specified

Continuation on the next page

Continuation of table 1

Study	Preprocessing	ML algorithm(s)	Best results	Further results
Chen, L. Zhou, and Yu (2021): RF NIDS	Cleaning; standardization; merging; removal of features with correlation > 0.95; Adaptive Synthetic Sampling; Random Under-Sampling; SMOTE	RF	Macro F1-Score = 0.93409; ROC-AUC = 0.99521	Minor improvements through sampling strategies: Adaptive Synthetic Sampling led to macro F1-Score = 0.95303
Fang and Xie (2025): Unsupervised/SSL-approach using open-set recognition	Cleaning; feature removal (irrelevant, all zero, redundant) to a total of 69	Information Maximizing Generative Adversarial Nets	Closed-set accuracy = 0.961	By fine-grained normal traffic subclassification precision rose to 0.993 for the closed-set and precision and recall for unknown traffic > 0.92
Kummerow et al. (2024): Transformer based AE	Encoding of some features; reduced packet structures	Transformer-based network traffic AE	F1-Score = 0.95; FPR = 0.045	Not specified
Maseer et al. (2021): Comparison of ML-NIDS-algorithms	Numericalization; normalization; use of 38 features; class balancing	Artificial NN; DT; k-Nearest-Neighbor; NB; RF; SVM; convolutional NN; k-means; self-organizing maps; expectation-maximization	No overall best performance, some algorithms classified perfectly on some attacks but missed others. Study does not provide average scores.	Many algorithms performed poorly for SQL Injection (0.0) (only 9 instances in testing set); performed better by improved class balance.
Mourouzis and Avgousti (n.d.): General NIDS evaluation	Not specified	J48 (DT); RF; IF; AdaBoost.M1	J48: Accuracy = 0.9998; FPR = 0.0002	IF: Performed poorly; Accuracy = 0.36567

Continuation on the next page

Continuation of table 1

Study	Preprocessing	ML algorithm(s)	Best results	Further results
Padhiyar and Tushyan (2025): Optimization using imbalance techniques	Data cleaning; PCA with 95% retaining variance; resampling techniques; ensemble methods	LR; SVM; DT	SVM: F1-Score = 0.73; AUC-ROC = 0.78	Hybrid sampling and RF as ensemble method, improved F1-Score to 0.90 and AUC-ROC to 0.94.
Roshan and Zafar (2021): Use of AE and Shapley Additive Explanation (SHAP)	StandardScaler; feature selection based Explainable Artificial Intelligence (xAI) techniques	AE	F1-Score = 0.90; AUC-ROC = 0.969 when using proposed xAI approach	The AE performed worse without the xAI techniques
Shyaa et al. (2023): Incremental Learning Genetic Programming Combiner	Not specified	Genetic programming combiner	Modified with online sequential extreme learning machine achieved an Accuracy = 0.90, F1-Score = 0.898, AUC = 0.903	Not specified
Verkerken et al. (2022): Different unsupervised approaches	Cleaning; duplicate removal (total of 67 features); normalization; PCA	IF; AE; One-Class SVM	AE: AUC-ROC = 0.9775; Accuracy = 0.9426; F1-Score = 0.9616	IF: AUC-ROC = 0.9584, Accuracy = 0.9111; F1-Score = 0.9391
Xu and Y. Liu (2025): Robust NIDS	StandardScaler	Multi-Layer Perceptron; 1D convolutional NN; One-Class SVM; Local Outlier Factor	Multi-Layer Perceptron: F1-Score = 0.9446 and Accuracy = 0.9975	Zero-Day attacks were simulated where One-Class SVM performed best, Multi-Layer-Perceptrons F1-Score dropped to 0.2937
Yuyang Zhou et al. (2020): Feature selection and ensemble classifier	CFS-BA for dimensionality reduction, transformation, normalization	Combination of C4.5, RF and Forest by Penalizing Attributes	Combination scored 0.999 for Accuracy, F1-Score and FPR = 0.001 for reduced features (13)	Ensemble scored worse when having the full 78 features

Continuation on the next page

Continuation of table 1

Study	Preprocessing	ML algorithm(s)	Best results	Further results
Yang Zhou et al. (2024): grouping and multi-AE	Normalization; feature grouping	Multi-AE	F1-Score = 0.7638; Accuracy = 0.8513; FPR = 0.1789	Metrics improved by an average of 10.64% through feature grouping

Table 1: Related Work

4 Experiment

This experiment implements three ML-approaches and trains and tests them on the CICIDS dataset. Afterwards, they are compared with each other and the related work in Chapter 3. Generally, the approaches consist of one established supervised, one established unsupervised, and one novel SSL algorithm. This design allows to evaluate the different training strategies. Furthermore, comparing simpler algorithms with a more complex, novel framework provides insights into the current A-NIDS development. In this chapter, the notes and assumptions, the general choice of algorithms, their working principles and implementations, the data preprocessing and the evaluation metrics are described.

4.1 Assumptions and Notes

Before explaining the experiment, some assumptions and notes have to be stated in regard to presented concepts in Chapter 2.

- **Full NIDS:** Regarding the four components of a NIDS, this paper only focuses on the preprocessor and the decision engine, the decoder and alert system are not considered or implemented. Thus, this paper recognizes that the results may not be applicable to real-world traffic due to the dataset creation procedure.
- **A-NIDS through ML:** In Section 2.2, three assumptions from Gates and Taylor (2007, p. 6) were presented which need to be considered for an A-NIDS. They are recognized and acknowledged as ML-A-NIDS provide unique advantages in the context of network security (Diana, Dini, and Paolini 2025, p. 35).
- **Dataset:** In this experiment, the CICIDS dataset is the only dataset used to evaluate the three different ML-approaches, no further datasets are referred to. As shown in Chapter 3, many papers conducted their research using CICIDS or tested their ML-algorithms using this dataset. Thus, the dataset allows for a meaningful comparison to other works in the field. The critique of Engelen, Rimmer, and Joosen (2021) in Section 2.4 is noted, but does not change the decision to use this dataset nor does this paper address or resolve the critique.
- **Semi-supervised learning:** To provide a complete overview, this ML-approach is presented in Section 2.3 but is neither implemented nor evaluated in this paper.

- **Contextual anomalies:** To provide a complete overview, this type of anomaly is presented in Section 2.2 but is not taken into consideration when implementing the ML-algorithms.
- **Reproducibility:** For reproducibility, random seeds were fixed to ensure reproducibility of dataset splitting, model initialization, and hyperparameter search.
- **Artificial Intelligence:** Python code was generated with the assistance of a large language model (OpenAI’s ChatGPT, GPT-5).
- **Hardware and Software setup:** To accelerate training, validation and testing, cloud resources from vast.ai were used including an NVIDIA H200 NVL with 140.4 GB VRAM and an AMD EPYC 9655 96-Core Processor with 322.4 GB RAM. The PyTorch Development Environment is selected.

4.2 Selected Algorithms

The algorithm selection for this experiment is based on several factors, though other algorithms may also perform well or be more suitable. As for the established supervised algorithm, the RF was selected. Géron (2017, p. 245–249) calls the RF one of the most powerful ML algorithms available and highlights the advantages of the underlying ensemble technique. Additionally, the RF has well-performed in the related work in Chapter 3 and is therefore chosen. In regard to the unsupervised algorithm, many novel frameworks classify their work as unsupervised but for this work, the selected approach should not be extraordinary but rather established. This results in the IF as this algorithm follows ensemble techniques and was specifically created for anomaly detection by F. T. Liu, Ting, and Z.-H. Zhou (2012). Interestingly, results seem to differ in regard to the related work of Mourouzis and Avgousti (n.d.) and Verkerken et al. (2022), thus, another implementation might be insightful. For the SSL-approach, the ET-SSL-framework was selected because it clearly states its algorithm as SSL. Additionally, it is one of the most recent papers found (published 22nd of July 2025) and addresses many state-of-the-art challenges of ML-A-NIDS.

4.2.1 Supervised Random Forest

This explanation of a RF is based on Géron (2017) and therefore no further citations are used. A RF is a collection of many DT. Instead of a single tree classifying an instance, many different trees decide and their combined results are used for the final classification. This approach is called ensemble learning with the underlying idea that a group outperforms a single predictor. Each DT is trained on a random subset of the training data, therefore each tree has a different dataset (called bootstrap aggregating). When a tree splits at a node, it does not consider all

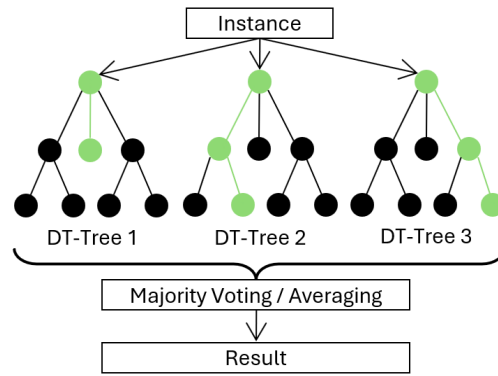


Figure 6: Conceptual illustration of a Random Forest

features but only takes a random subset of features into consideration (called random feature selection). Thus, the trees are less correlated and therefore the RF stronger. There are two possible outputs for a tree. Either each tree classifies the instance and the forest selects the class with the most votes or each tree predicts an anomaly score and the forest returns the average score. The final classification is controlled by a threshold δ . Formally, this means for the first decision type:

$$\hat{y} = \arg \max_{c \in C} \sum_{j=1}^m I(h_j(x) = c)$$

$h_j(x)$ is the prediction of the j -th tree for an instance x , $I(\cdot)$ is the indicator function with an output of 1 or 0. For the second decision type:

$$\hat{y} = \frac{1}{m} \sum_{j=1}^m h_j(x).$$

The concept of a RF is shown in Figure 6. There are hyperparameters for a RF which lead to different results. A DT can have a maximum depth or a maximum number of leaves. The sample size for the bootstrap aggregation and random feature selection may differ. And finally, the number of trees m is variable. The threshold δ can be optimized as well. All hyperparameters are optimized in the validation process. Scikit-learn's `RandomForestClassifier` was used to implement the RF.

4.2.2 Unsupervised Isolation Forest

The IF is an unsupervised ML-approach which fully relies on the assumption that anomalies are few and different in regard to the normal instances in a dataset. Note that this entire subsection is based on the paper of F. T. Liu, Ting, and Z.-H. Zhou (2012) and therefore no further citations

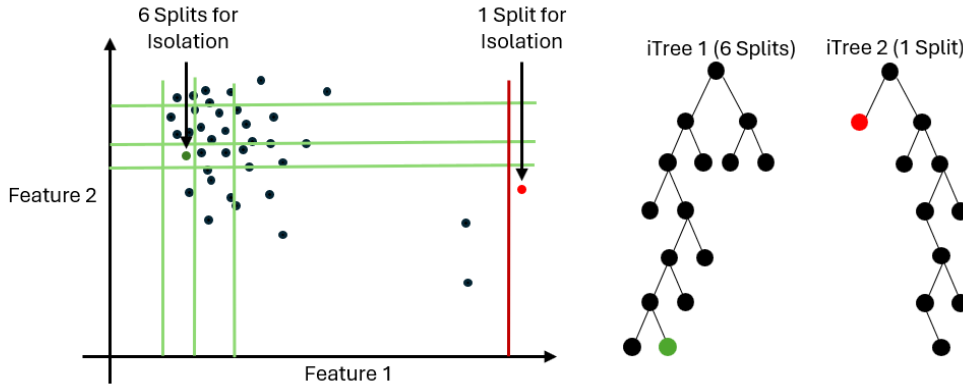


Figure 7: Conceptual illustration of an Isolation Forest

are used. An IF is a collection of many isolation trees, thus an IF is based on ensemble learning as well. Many isolation trees provide an anomaly score for a single instance, the IF then takes the mean of these scores. An isolation tree recursively partitions the dataset by selecting a feature $q \in 1, \dots, d$ and a split value p between the minimum and the maximum of the feature q . This continues until the tree reaches a single instance of the dataset and thus, has achieved a pure partition. The path length $h(x)$ in an isolation tree states the number of edges an instance x traversed until it is isolated. In clusters of instances, it usually requires more splits to isolate them, hence the $h(x)$ is larger. An anomaly is isolated earlier leading to a smaller $h(x)$. The overall concept of an IF is shown in Figure 7. To lastly decide whether the instance is considered normal or anomalous, an anomaly score $s(x, n)$ for the instance x in a dataset with n instances is calculated:

$$s(x, n) = 2^{-\frac{E[h(x)]}{c(n)}}$$

$E[h(x)]$ is the average path length of x across all isolation trees. $c(n)$ is the average path length for an unsuccessful search in a binary search tree:

$$c(n) = 2H(n-1) - \frac{2(n-1)}{n}, \quad H(i) = \sum_{k=1}^i \frac{1}{k} \approx \ln(i) + \gamma$$

γ is the Euler-Mascheroni constant and approximately 0.5772156649. Overall, a number of isolation trees t is chosen, where each tree contributes to the IF. The IF's outcome, the anomaly score, then leads to the instance's classification as normal or anomalous based on a threshold δ . Further hyperparameters can be the maximum tree depth or the subsample size. The hyperparameters are optimized in the validation process. Scikit-learn's IsolationForest is used to implement the IF.

4.2.3 Encrypted Traffic Anomaly Detection using Self-supervised Contrastive Learning

The novel ET-SSL framework is proposed by Sattar et al. (2025) to face the challenges of end-to-end encrypted traffic, the high cost of data, and the ever-changing nature of network traffic and zero-day attacks. Note that this entire subsection is based on this paper and therefore no further citations are used. The overall approach relies on contrastive SSL to create an optimized feature space with well-separated normal and anomalous embeddings. Even though the paper only uses five flow-level features, in this work a different data preprocessing is chosen with regard to the best practices found in literature. Each flow x_i of the CICIDS dataset is represented as a vector in a high-dimensional feature space. x_i is projected into a lower-dimensional embedding space by employing a neural encoder $f_\theta(\cdot)$ resulting in $z_i = f_\theta(x_i)$. The contrastive learning procedure described in Section 2.3 is realized through data augmentation $g(\cdot)$. By noise injection and scaling, a positive pair (x_i, x_i^+) is created. The contrastive loss ($L_{\text{contrastive}}$) ensures that semantically similar traffic embeddings stay close, and dissimilar ones are pushed apart. The concept of the contrastive learning and embedding space creation is shown in Figure 8. The loss can be defined by this function:

$$L_{\text{contrastive}} = -\frac{1}{n} \sum_{i=1}^n \log \frac{\exp(\text{sim}(z_i, z_i^+)/\tau)}{\sum_{j=1}^n \exp(\text{sim}(z_i, z_j)/\tau)}.$$

The cosine similarity $\text{sim}(z_a, z_b)$ measures the similarity between two embeddings. The temperature parameter τ sets the sharpness of the similarity scores. The anomaly detection works through an anomaly score calculation which measures the distance from the centroid (mean of all normal traffic embeddings) to the instance:

$$S(t_i) = \|z_i - \mu_{\text{norm}}\|^2, \quad \mu_{\text{norm}} = \frac{1}{|N|} \sum_{i \in N} z_i.$$

The final decision is made by the threshold θ , which is optimized via validation:

$$A(t_i) = \begin{cases} 1 & \text{if } S(t_i) > \theta, \\ 0 & \text{otherwise.} \end{cases}$$

Note that the presented framework distinguishes between the decision threshold θ and the intra-normal compactness margin δ , even though in most works the decision threshold is denoted as δ . To refine the separation of normal and anomalous embeddings, an anomaly loss is used (if the classifier's output $A(t_i)$ equals 1):

$$L_{\text{total}} = L_{\text{contrastive}} + \gamma L_{\text{anomaly}}, \quad L_{\text{anomaly}} = \sum_{i=1}^n I(A(t_i)) \cdot \|z_i - z_0\|^2.$$

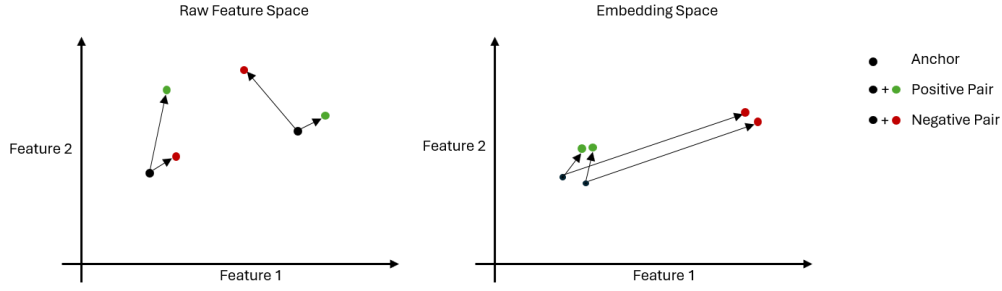


Figure 8: Contrastive learning process and Embedding Space creation

To face the challenge of zero-day attack detection, not only the distance to the normal centroid is taken into consideration but also the distance to the anomalous centroid:

$$S(d_i) = \|z_i - \mu_{\text{norm}}\|^2 - \kappa \cdot \|z_i - \mu_{\text{anom}}\|^2.$$

Lastly, this framework aims to adapt to evolving traffic and thus updates the centroid of normal traffic incrementally. To achieve this, a weighted average of the centroid at time t and the new centroid at $t + 1$ is taken. The weight depends on the factor α , where an α closer to 1 results in slower adaptation and closer to 0 in faster adaptation of the new centroid:

$$\mu_{\text{norm}}^{(t+1)} = \alpha \mu_{\text{norm}}^{(t)} + (1 - \alpha) \cdot \frac{1}{|N|} \sum_{i \in N} z_i.$$

ET-SSL has two important hyperparameters (τ, θ) which are optimized in the validation process. If a margin constraint is used, δ denotes the intra-normal compactness margin (distinct from the decision threshold θ).

4.3 Data Preprocessing

In regard to the best practices and procedures found in the related work, the dataset is preprocessed. This is realized by the following steps:

1. All instances with missing values are removed to facilitate further work.
2. Categorical features are removed because the ET-SSL-approach relies on distance measuring.
3. Constant features are removed via VarianceThreshold.
4. All features are normalized using the Scikit-Learn's MinMaxScaler during model training (scalers are part of the pipelines).

5. The highly imbalanced CICIDS-dataset requires additional preprocessing. For the RF, imbalance in the training data is mitigated by using `class_weight='balanced_subsample'`. Over-sampling in both hyperparameter search and final training is avoided to reduce runtime. The decision threshold δ is tuned on a validation split. Additionally, stratified subsampling was applied during cross-validation, and for runtime safety, an upper limit on the maximum tree depth was enforced even if cross-validation suggested no limit. For the unsupervised IF and the ET-SSL, no resampling is applied, as both rely on the assumption of anomalous classes being rare and distinct.
6. The training, validation and testing datasets are split using a 70/15/15 ratio. For supervised learning, the splits are stratified to ensure that the class distribution is preserved across the datasets. Both benign and malicious traffic is used but in a binary form. For the unsupervised IF, only benign traffic is used in the training stage, while validation and test data contain both benign and malicious traffic. For the ET-SSL, both benign and malicious traffic (binary) are included to allow the contrastive representation learning.
7. To reduce redundancy and computational complexity, highly correlated features with a Pearson correlation coefficient greater than 0.95 are combined using Principal Component Analysis (PCA) for the RF and IF keeping 0.99 variance. 0.95 is selected as it has been widely used in previous work with good results. For the ET-SSL, dimensionality reduction is directly handled by the encoder.
8. For the ET-SSL-approach, additional stabilization measures were applied beyond the original paper: the encoder was extended with Batch Normalization and Dropout layers, stronger augmentations were used (including feature dropout), and gradient clipping as well as early stopping were introduced to improve convergence and avoid overfitting. The optimization of thresholds was performed using a wider range of κ values.

4.4 Validation and Metrics

After training the models on 70% of the data, they are validated against a held-out 15% split, while the final 15% is used for testing. The macro F1-Score is chosen as the main optimization criterion since it balances precision and recall, which are the most critical metrics in the context of highly imbalanced datasets as discussed in Section 2.3. Hyperparameter selection and decision thresholds are tuned based on the validation set.

- For the supervised RF, a randomized hyperparameter search is conducted using stratified 3-fold cross-validation on the training split. Parameters such as maximum tree depth, minimum samples per leaf, and feature sampling ratio are explored. During cross-validation, a

fixed 0.5 threshold was used for efficiency. The decision threshold δ was tuned afterwards on the validation split to maximize macro F1-score.

- For the unsupervised IF, hyperparameters including the number of trees, subsample size, and contamination level are varied in a small grid. The model is trained without labels, but validation labels are used to determine the optimal anomaly score threshold δ by maximizing the macro F1-Score on the validation split.
- For the self-supervised ET-SSL, randomized hyperparameter combinations of the margin τ , batch size b , and decision threshold θ are sampled. After contrastive pre-training, validation macro F1-Score guides both the choice of hyperparameters and the final threshold θ . Due to the computational cost of training, a limited randomized search was applied.

To evaluate the performance of the implemented algorithms and to allow a comparison to the related work, the metrics presented in Section 2.3 are used. This includes:

- The overall classification results in the form of a confusion matrix.
- TPR, TNR, FPR, False Negative Rate (FNR)
- Accuracy
- ROC-AUC
- Precision-Recall-AUC
- F1-Score
- Training time
- Inference time

In the context of NIDS and the imbalanced CICIDS dataset, the macro (not weighted) metrics are used, to properly assess the performance for the small attack classes. To evaluate the model's performance on zero-day attacks, all the steps described in the previous sections are repeated but the second time without training and validating on three attack types. Bots and Web Attack — Brute Force are chosen to represent diverse attack procedures and as a third type, infiltration is selected because it is highly underrepresented in the dataset with only 36 instances in total.

5 Results

In this experiment, three algorithms were trained on the CICIDS dataset by optimizing thresholds and hyperparameter in base and a zero-day scenario. This chapter evaluates their performance based on the metrics explained in Section 4.4. Additionally, a brief dataset analysis is provided.

5.1 Exploratory Dataset Analysis

The CICIDS dataset contains 2,830,743 rows and 79 features. Rows with missing values were removed. Table 2 provides an overview of all classes and the total number of instances for each one. All features are shown in Table 3 in the Appendix.

Label	Count
BENIGN	2273097
DoS Hulk	231073
PortScan	158930
DDos	128027
DoS GoldenEye	10293
FTP — Patator	7938
SSH — Patator	5897
DoS slowloris	5796
DoS Slowhttptest	5499
Bot	1966
Web Attack — Brute Force	1507
Web Attack — XSS	652
Infiltration	36
Web Attack — Sql Injection	21
Heartbleed	11

Table 2: CIC-IDS2017 Class Distribution

5.2 Findings

Across both scenarios the RF is clearly strongest, achieving near-perfect discrimination and extremely low error rates. ET-SSL ranks second with robust results that degrade only moderately under zero-day conditions, while IF trails significantly in recall and average precision.

In the base scenario, RF reaches a macro F1-Score of 0.9942 with ROC-AUC 1.0000 and precision-recall 0.9999. Its TPR is 1.0000 against a TNR of 0.9954, corresponding to FPR = 0.0046 and essentially FNR ≈ 0 (4 misses on 83,458 positives), for an overall accuracy of 0.9963. The tuned probability threshold is very small ($\delta \approx 0.0095$), indicating confident separation. Applying the trained RF to the test split took about 1.18 s in the chosen environment.

ET-SSL attains a macro F1-Score of 0.8611 (ROC-AUC 0.9477, precision-recall 0.8214). Compared with the RF, the boundary is looser: TPR 0.8283 and TNR 0.9278, FPR 0.0722 and FNR 0.1717, yielding accuracy 0.9082. The best distance threshold is close to zero ($\theta \approx 8 \cdot 10^{-4}$), which is consistent with squared-distance scoring in the embedding space; inference is effectively instantaneous for the trained encoder.

IF lags behind with macro F1-Score 0.7069 (ROC-AUC 0.7296, precision-recall 0.4739). Its recall is the key weakness (TPR 0.4384, FNR 0.5616), while TNR remains 0.9333 (FPR 0.0667); accuracy is 0.8359 with a tuned score threshold near 0.5. The results suggest that CICIDS' heterogeneous attacks violate IF's assumptions (rare, well-separated anomalies) when used without additional representation learning or hybridization.

Under the zero-day split the overall picture remains stable. The RF degrades only slightly to macro F1-Score 0.9931 (ROC-AUC 0.9989, precision-recall 0.9982). Error rates change slightly to TPR 0.9967, TNR 0.9953 (FPR 0.0047, FNR 0.0033), accuracy 0.9956 and $\delta \approx 0.0033$; inference time is 1.24 s. ET-SSL drops moderately to macro F1-Score 0.8512 (ROC-AUC 0.9248, precision-recall 0.6530) with TPR 0.8098, TNR 0.9243 (FPR 0.0757, FNR 0.1902), accuracy 0.9018 and $\theta \approx 2 \cdot 10^{-4}$. IF remains essentially unchanged (macro F1-Score 0.7069), continuing to miss more than half of positives.

5.3 Closed-Set Results (Base Scenario)

Figure 9 shows the confusion matrices for all three models: the RF matrix is dominated by the main diagonal, with only four false negatives and very few false positives. This aligns with the nearly rectangular ROC and the high-lying PR curves in Figure 10. ET-SSL produces a clearly better balance than IF, but with a visibly higher false-negative block (consistent with FNR $\approx 17\%$). IF's matrix shows under-detection of the positive class, which is also reflected by a low PR curve and modest ROC area.

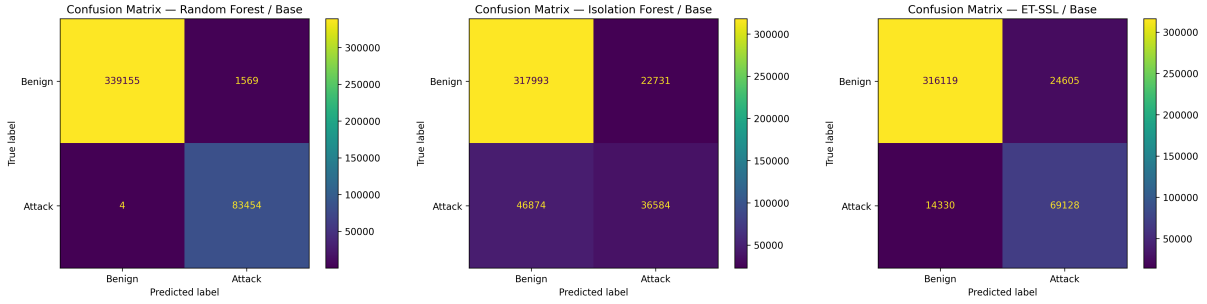


Figure 9: Confusion matrices (base scenario): RF (left), IF (center), ET-SSL (right).

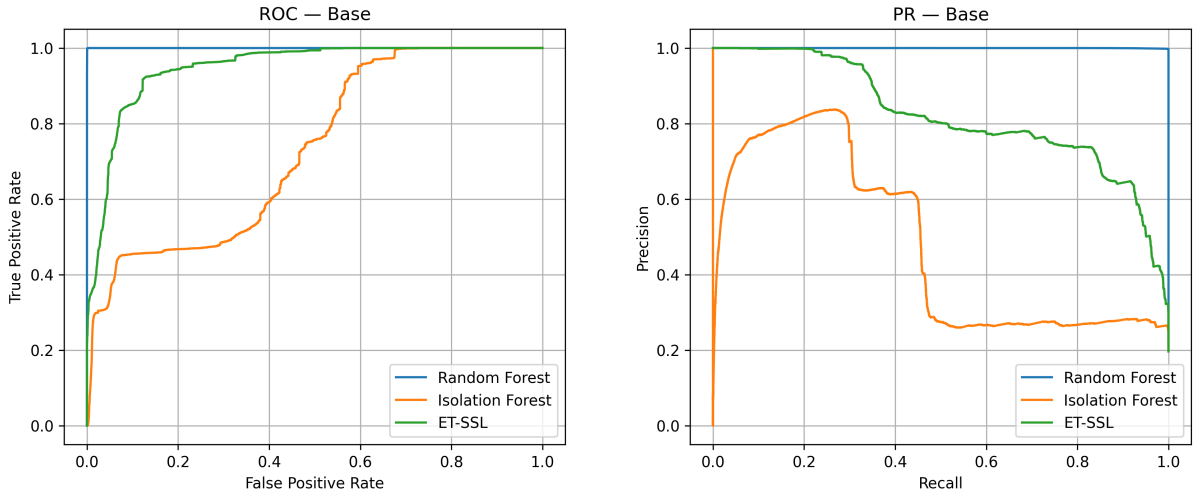


Figure 10: ROC (left) and PR (right) for the base scenario.

In practice, the base-scenario results imply that the RF can be deployed with a very conservative threshold to keep false negatives near zero while sustaining a sub-percent false-positive rate, minimizing analyst load. ET-SSL offers a thresholdable anomaly score and compact embedding that could be leveraged for downstream triage, but would require tolerance for higher FPR and FNR than the RF. IF, in its plain form, is not suitable as a stand-alone detector on CICIDS.

5.4 Open-Set Results (Zero-Day Scenario)

Holding out *Bot*, *Web Attack — Brute Force* and *Infiltration* during training and validation stresses generalization to unseen attack morphologies. The RF remains stable: its confusion matrix in Figure 11 changes only marginally, and the ROC/PR curves in Figure 12 remain close to their base-scenario counterparts. This suggests that the forest partitions are driven by generic, transferable statistics (e.g., volumetric or temporal dynamics) that are not specific to the withheld families.

ET-SSL exhibits moderate degradation: macro F1-score decreases by about one percentage point and, more notably, precision-recall falls from ≈ 0.82 to ≈ 0.65 , indicating that score

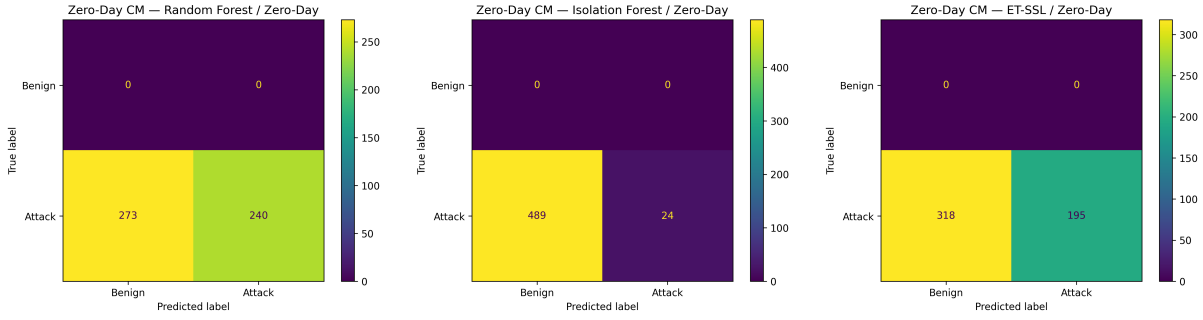


Figure 11: Confusion matrices (zero-day scenario): RF (left), IF (center), ET-SSL (right).

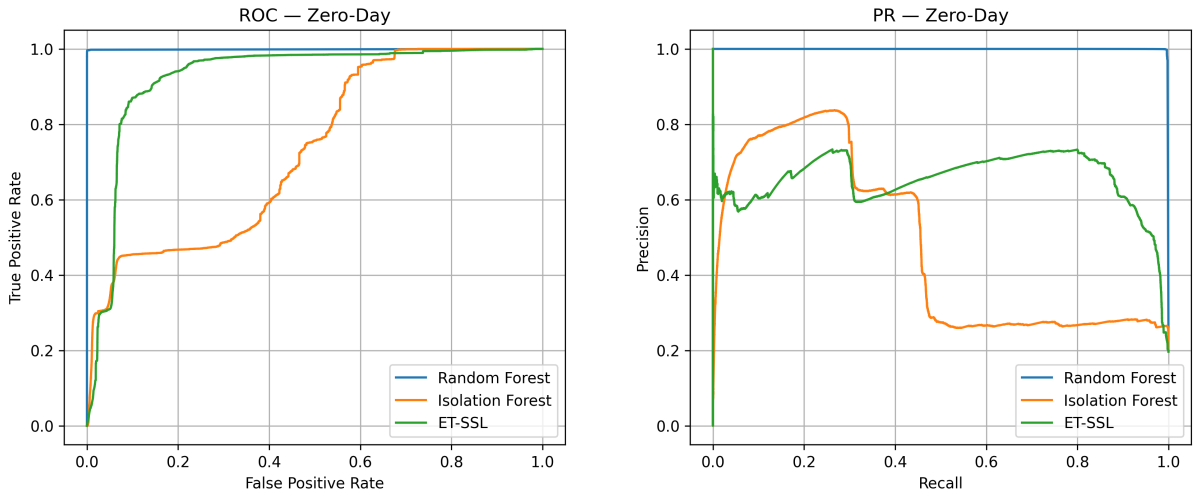


Figure 12: ROC (left) and PR (right) for the zero-day scenario.

ranking for the long tail of novel positives becomes harder. Nevertheless, ET-SSL still separates benign and malicious traffic sufficiently well to remain useful in an open-set setting, clearly outperforming IF. IF’s performance barely moves but remains recall-limited, confirming that simple isolation heuristics are brittle in this dataset without additional feature learning.

5.5 Discussion

In Summary, the results show that a carefully tuned supervised baseline on CICIDS can reach state-of-the-art performance and retain it under a zero-day split, with almost no loss in F1 or AP and with both FNR and FPR at or below half a percent. This confirms the findings of Chen, L. Zhou, and Yu (2021) and Al Lail, Garcia, and Olivo (2023), where RF achieved macro F1-scores above 0.93, and even exceeds them due to additional preprocessing and threshold tuning.

In contrast, the IF shows the risks of purely unsupervised anomaly detection: its assumption of “few and well-separated” anomalies is violated by CICIDS’ heterogeneous and overlapping attack types. This explains why more than half of malicious instances are missed, consistent

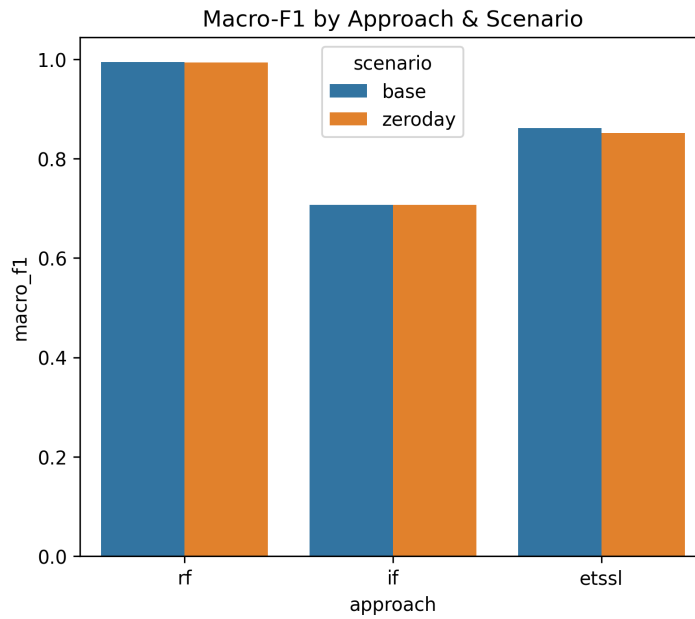


Figure 13: Macro-F1 overview by approach and scenario.

with the poor IF results reported by Mourouzis and Avgousti (n.d.). Studies that achieved stronger IF performance (e.g., Verkerken et al. (2022)) relied heavily on preprocessing and feature engineering, which were deliberately kept limited here to ensure comparability across models.

ET-SSL provides a principled embedding and anomaly score and works better than IF, but its performance remains below both its original report (Sattar et al. 2025) and unsupervised deep autoencoder frameworks such as Carrera et al. (2022). Two factors likely contribute: first, the strong class imbalance and noisy minority classes in CICIDS increase the false-negative rate; second, the generic contrastive setup struggles with highly heterogeneous traffic features. As related work suggests, additional open-set losses (Fang and Xie 2025) or hybrid semi-supervised refinements could address these limitations.

RF is the most suitable of the three for a production NIDS on CICIDS-like traffic, offering an excellent miss/false-alarm trade-off and fast inference. ET-SSL is attractive as an auxiliary model for adaptive scoring and triage, particularly where labeled data are scarce or drift is expected. IF, however, is not recommended as a stand-alone detector in this setting unless combined with stronger feature learning or hybrid ensemble strategies.

6 Conclusion

This chapter summarizes the findings of the experiment, answers the research questions, and highlights avenues for future research.

6.1 Summary

The comparative evaluation of a supervised RF, an unsupervised IF, and the novel self-supervised ET-SSL framework on the CICIDS dataset revealed clear performance differences. The RF achieved near-perfect results in both closed-set and zero-day scenarios, confirming the robustness of supervised ensemble methods when sufficient labels are available. In contrast, ET-SSL produced competitive but less stable results, showing resilience under zero-day conditions but suffering from increased false negatives and lower precision recall. The IF, meanwhile, struggled significantly, reflecting the limitations of density-based anomaly detection in heterogeneous and imbalanced datasets.

Compared to related work, the RF results not only confirm but also exceed prior findings, while the challenges observed with IF and ET-SSL align with studies highlighting the difficulties of unsupervised and SSL methods without extensive preprocessing or hybridization. Taken together, these results underline that supervised learning currently offers the most reliable strategy for anomaly-based NIDS, while self-supervised approaches provide promising potential in scenarios with scarce labels or evolving traffic. Purely unsupervised methods, however, appear insufficient in their raw form.

6.2 Answer to the Research Questions

The guiding research question asked: *How do an established supervised, an unsupervised, and a novel self-supervised machine learning approach perform in network anomaly detection on the CICIDS dataset?* The study provides the following answers:

- **RQ2:** The RF consistently outperforms both alternatives, reaching macro F1 ≈ 0.99 , ROC-AUC ≈ 1.0 , and almost no false negatives. ET-SSL performs moderately (macro F1 ≈ 0.85), while IF lags behind (macro F1 ≈ 0.71).

- **RQ3:** Inference times are negligible for ET-SSL and IF, while the RF requires about one second per test split but remains operationally feasible.
- **RQ4:** The RF generalizes best to unseen attacks with virtually no drop in performance. ET-SSL degrades moderately, while IF continues to miss more than half of the anomalies.
- **RQ5:** For deployment, RF is highly practical in labeled environments. ET-SSL offers potential in label-scarce or dynamic scenarios but requires refinement. IF in its plain form is unsuitable.

6.3 Future Research

The CICIDS dataset does not perfectly reflect real-world traffic and may introduce biases. Furthermore, only three algorithms were tested. Future research should therefore:

- Investigate the effects of label noise, class imbalance, and feature redundancies on model performance.
- Explore hybrid approaches, e.g., combining supervised classifiers with SSL-based embeddings to improve zero-day robustness.
- Develop open-set adaptation strategies such as novel loss functions or incremental learning to stabilize precision-recall under unseen attack classes.
- Examine real-time deployment challenges, including latency, drift, and interpretability in production-grade NIDS.

In conclusion, the Random Forest emerges as the most robust and reliable choice, ET-SSL shows promise under certain conditions, and IF proves unsuitable in its basic form. These findings provide a foundation for more adaptive, interpretable, and practically deployable security solutions.

Bibliography

- Abdulhammed, Razan et al. (2019). “Features Dimensionality Reduction Approaches for Machine Learning Based Network Intrusion Detection”. In: *Electronics* 8, pp. 2–27. DOI: [10.3390/electronics8030322](https://doi.org/10.3390/electronics8030322).
- Al Lail, Mustafa, Alejandro Garcia, and Saul Olivo (2023). “Machine Learning for Network Intrusion Detection—A Comparative Study”. In: *Future Internet* 15.7. DOI: [10.3390/fi15070243](https://doi.org/10.3390/fi15070243).
- Alotaibi, Fahad and Sergio Maffei (2024). “Mateen: Adaptive Ensemble Learning for Network Anomaly Detection”. In: *Proceedings of the 27th International Symposium on Research in Attacks, Intrusions and Defenses*. Ed. by Eleonora Losiouk et al. ACM Digital Library. Erscheinungsort nicht ermittelbar: Association for Computing Machinery, pp. 215–234. ISBN: 9798400709593. DOI: [10.1145/3678890.3678901](https://doi.org/10.1145/3678890.3678901). URL: <https://www.doc.ic.ac.uk/~maffei/papers/raid24.pdf#:~:text=We%20evaluate%20the%20effectiveness%20of,of%20the%20incoming%20samples>.
- Buck, Thomas (2020). “Anomaly Detection in Data Network Traffic with Machine Learning”. Master Thesis. Darmstadt: Hochschule Darmstadt. URL: https://fbmn.h-da.de/fileadmin/Dokumente/Studium/DS/2021_MDS_BuckThomas_THE.pdf.
- Cambridge Dictionary (2025). *Anomaly*. Ed. by Cambridge Dictionary. URL: <https://dictionary.cambridge.org/dictionary/english/anomaly>.
- Canadian Institute for Cybersecurity (2018). *Intrusion detection evaluation dataset (CIC-IDS2017)*. URL: <https://www.unb.ca/cic/datasets/ids-2017.html>.
- Carrera, Francesco et al. (2022). “Combining Unsupervised Approaches for Near Real-Time Network Traffic Anomaly Detection”. In: *Applied Sciences* 12.3. DOI: [10.3390/app12031759](https://doi.org/10.3390/app12031759).
- Chen, Zhewei, Linyue Zhou, and Wenwen Yu (2021). “ADASYN–Random Forest Based Intrusion Detection Model”. In: *2021 4th International Conference on Signal Processing and Machine Learning*. New York, NY, USA: ACM, pp. 152–159. ISBN: 9781450390170. DOI: [10.1145/3483207.3483232](https://doi.org/10.1145/3483207.3483232).
- Diana, Lorenzo, Pierpaolo Dini, and Davide Paolini (2025). “Overview on Intrusion Detection Systems for Computers Networking Security”. In: *Computers* 14, pp. 1–44. DOI: [10.3390/computers14030087](https://doi.org/10.3390/computers14030087).
- Engelen, Gints, Vera Rimmer, and Wouter Joosen (2021). “Troubleshooting an Intrusion Detection Dataset: the CICIDS2017 Case Study”. In: *2021 IEEE Symposium on Security and Privacy workshops*. Piscataway, NJ: IEEE. ISBN: 978-1-6654-3732-5. DOI: [10.1109/SPW53761.2021.00009](https://doi.org/10.1109/SPW53761.2021.00009).

- ENISA (2024). *ENISA Threat Landscape 2024*. Ed. by European Union Agency for Cybersecurity. URL: https://securitydelta.nl/media/com_hsd/report/690/document/ENISA-Threat-Landscape-2024.pdf.
- Fang, Jun and Cunxiang Xie (2025). “Unknown intrusion traffic detection method based on unsupervised learning and open-set recognition”. In: *Scientific reports* 15.1, p. 17001. DOI: [10.1038/s41598-025-01084-1](https://doi.org/10.1038/s41598-025-01084-1).
- Gates, Carrie and Carol Taylor (2007). “Challenging the anomaly detection paradigm”. In: *Proceedings of the 2006 workshop on New security paradigms*. Ed. by Abe Singer. ACM Other conferences. New York, NY: ACM, pp. 21–29. ISBN: 9781595939234. DOI: [10.1145/1278940.1278945](https://doi.org/10.1145/1278940.1278945).
- Géron, Aurélien (2017). *Hands-on machine learning with Scikit-Learn and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. First edition. Sebastopol: O’Reilly Media. ISBN: 978-1-491-96229-9. URL: <http://shop.oreilly.com/product/0636920052289.do>.
- Gharib, Amirhossein et al. (2016). “An Evaluation Framework for Intrusion Detection Dataset”. In: *ICISS 2016*. Piscataway, NJ: IEEE, pp. 1–6. ISBN: 978-1-5090-5493-0. DOI: [10.1109/ICISSEC.2016.7885840](https://doi.org/10.1109/ICISSEC.2016.7885840).
- Goldstein, Markus and Seiichi Uchida (2016). “A Comparative Evaluation of Unsupervised Anomaly Detection Algorithms for Multivariate Data”. In: *PloS one* 11.4. DOI: [10.1371/journal.pone.0152173](https://doi.org/10.1371/journal.pone.0152173).
- International Telecommunication Union (2024). *Measuring digital development. Facts and Figures 2024*. Ed. by ITU. URL: https://www.itu.int/dms_pub/itu-d/opb/ind/d-ind-ict_mdd-2024-4-pdf-e.pdf.
- Kohli, Mudita and Indu Chhabra (2025). “A comprehensive survey on techniques, challenges, evaluation metrics and applications of deep learning models for anomaly detection”. In: *Discover Applied Sciences* 7.7. DOI: [10.1007/s42452-025-07312-7](https://doi.org/10.1007/s42452-025-07312-7).
- Kummerow, André et al. (2024). “Unsupervised Anomaly Detection and Explanation in Network Traffic with Transformers”. In: *Electronics* 13.22, p. 4570. DOI: [10.3390/electronics13224570](https://doi.org/10.3390/electronics13224570).
- Liu, Fei Tony, Kai Ming Ting, and Zhi-Hua Zhou (2012). “Isolation-Based Anomaly Detection”. In: *ACM Transactions on Knowledge Discovery from Data* 6.1, pp. 1–44. ISSN: 1556-4681. DOI: [10.1145/2133360.2133363](https://doi.org/10.1145/2133360.2133363).
- Maseer, Ziadoon Kamil et al. (2021). “Benchmarking of Machine Learning for Anomaly Based Intrusion Detection Systems in the CICIDS2017 Dataset”. In: *IEEE Access* 9, pp. 22351–22370. DOI: [10.1109/ACCESS.2021.3056614](https://doi.org/10.1109/ACCESS.2021.3056614).
- Mourouzis, Theodosios and Andreas Avgousti (n.d.). *Intrusion Detection with Machine Learning Using Open-Sourced Datasets*. URL: <https://arxiv.org/pdf/2107.12621>.
- Oyelakin, Akinyemi et al. (2023). “Overview and Exploratory Analyses of CICIDS 2017 Intrusion Detection Dataset”. In: *Journal of Systems Engineering and Information Technology (JOSEIT)* 2.2, pp. 45–52. DOI: [10.29207/joseit.v2i2.5411](https://doi.org/10.29207/joseit.v2i2.5411).

- Padhiyar, Mehul Kumar and Vikas N. Tulshyan (2025). “Optimizing Ids Performance With Class Imbalance Techniques”. In: *International Journal of Creative Research Thoughts (IJRCT)* 13.3, pp. 421–431. URL: <https://www.ijcrt.org/papers/IJCRT2503982.pdf>.
- Roshan, Khushnaseeb and Aasim Zafar (2021). “Utilizing XAI Technique to Improve Autoencoder based Model for Computer Network Anomaly Detection with Shapley Additive Explanation(SHAP)”. In: *International journal of Computer Networks & Communications* 13.6, pp. 109–128. ISSN: 09752293. DOI: [10.5121/ijcnc.2021.13607](https://doi.org/10.5121/ijcnc.2021.13607). URL: <http://arxiv.org/pdf/2112.08442>.
- Sattar, Sadaf et al. (2025). “Anomaly detection in encrypted network traffic using self-supervised learning”. In: *Scientific reports* 15. DOI: [10.1038/s41598-025-08568-0](https://doi.org/10.1038/s41598-025-08568-0).
- Sharafaldin, Iman, Arash Habibi Lashkari, and Ali A. Ghorbani (2018). “Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization”. In: *Proceedings of the 4th International Conference on Information Systems Security and Privacy*. SCITEPRESS - Science and Technology Publications, pp. 108–116. ISBN: 978-989-758-282-0. DOI: [10.5220/0006639801080116](https://doi.org/10.5220/0006639801080116).
- Shyaa, Methaq A. et al. (2023). “Enhanced Intrusion Detection with Data Stream Classification and Concept Drift Guided by the Incremental Learning Genetic Programming Combiner”. In: *Sensors (Basel, Switzerland)* 23.7. DOI: [10.3390/s23073736](https://doi.org/10.3390/s23073736).
- Tanenbaum, Andrew S. (2011). *Computer networks*. 5th ed. Boston: Prentice Hall. ISBN: 9780133485936. URL: <https://learning.oreilly.com/library/view/-/9780133485936/?ar>.
- Verkerken, Miel et al. (2022). “Towards Model Generalization for Intrusion Detection: Unsupervised Machine Learning Techniques”. In: *Journal of Network and Systems Management* 30.1. ISSN: 1064-7570. DOI: [10.1007/s10922-021-09615-7](https://doi.org/10.1007/s10922-021-09615-7).
- Xu, Zhaoyang and Yunbo Liu (2025). *Robust Anomaly Detection in Network Traffic: Evaluating Machine Learning Models on CICIDS2017*. URL: <http://arxiv.org/pdf/2506.19877v1>.
- Zhou, Yang et al. (2024). “Network traffic anomaly detection model based on feature grouping and multi-autoencoders integration”. In: *Electronics Letters* 60.23. ISSN: 0013-5194. DOI: [10.1049/el12.70103](https://doi.org/10.1049/el12.70103).
- Zhou, Yuyang et al. (2020). “Building an efficient intrusion detection system based on feature selection and ensemble classifier”. In: *Computer Networks* 174, p. 107247. ISSN: 13891286. DOI: [10.1016/j.comnet.2020.107247](https://doi.org/10.1016/j.comnet.2020.107247).

A Appendix

Feature	MissingCount	MissingRate
Flow_Packets_s	2867	0.1013
Flow_Bytes_s	2867	0.1013
Destination_Port	0	0
Average_Packet_Size	0	0
Fwd_Avg_Bulk_Rate	0	0
Fwd_Avg_Packets_Bulk	0	0
Fwd_Avg_Bytes_Bulk	0	0
Fwd_Header_Length_1	0	0
Avg_Bwd_Segment_Size	0	0
Avg_Fwd_Segment_Size	0	0
Down_Up_Ratio	0	0
Bwd_Avg_Packets_Bulk	0	0
ECE_Flag_Count	0	0
CWE_Flag_Count	0	0
URG_Flag_Count	0	0
ACK_Flag_Count	0	0
PSH_Flag_Count	0	0
RST_Flag_Count	0	0
SYN_Flag_Count	0	0
FIN_Flag_Count	0	0
Bwd_Avg_Bytes_Bulk	0	0
Bwd_Avg_Bulk_Rate	0	0
Packet_Length_Std	0	0
Subflow_Fwd_Packets	0	0
Idle_Min	0	0
Idle_Max	0	0
Idle_Std	0	0
Idle_Mean	0	0
Active_Min	0	0
Active_Max	0	0
Active_Std	0	0
Active_Mean	0	0

Continuation on the next page

Continuation of table 3

Feature	MissingCount	MissingRate
min_seg_size_forward	0	0
act_data_pkt_fwd	0	0
Init_Win_bytes_backward	0	0
Init_Win_bytes_forward	0	0
Subflow_Bwd_Bytes	0	0
Subflow_Bwd_Packets	0	0
Subflow_Fwd_Bytes	0	0
Packet_Length_Variance	0	0
Max_Packet_Length	0	0
Packet_Length_Mean	0	0
Bwd_Packet_Length_Max	0	0
Flow_IAT_Min	0	0
Flow_IAT_Max	0	0
Flow_IAT_Std	0	0
Flow_IAT_Mean	0	0
Bwd_Packet_Length_Std	0	0
Bwd_Packet_Length_Mean	0	0
Bwd_Packet_Length_Min	0	0
Fwd_Packet_Length_Std	0	0
Flow_Duration	0	0
Fwd_Packet_Length_Mean	0	0
Fwd_Packet_Length_Min	0	0
Fwd_Packet_Length_Max	0	0
Total_Length_of_Bwd_Packets	0	0
Total_Length_of_Fwd_Packets	0	0
Total_Backward_Packets	0	0
Total_Fwd_Packets	0	0
Fwd_IAT_Total	0	0
Fwd_IAT_Mean	0	0
Fwd_IAT_Std	0	0
Fwd_IAT_Max	0	0
Min_Packet_Length	0	0
Bwd_Packets_s	0	0
Fwd_Packets_s	0	0
Bwd_Header_Length	0	0
Fwd_Header_Length	0	0
Bwd_URG_Flags	0	0
Fwd_URG_Flags	0	0

Continuation on the next page

Continuation of table 3

Feature	MissingCount	MissingRate
Bwd_PSH_Flags	0	0
Fwd_PSH_Flags	0	0
Bwd_IAT_Min	0	0
Bwd_IAT_Max	0	0
Bwd_IAT_Std	0	0
Bwd_IAT_Mean	0	0
Bwd_IAT_Total	0	0
Fwd_IAT_Min	0	0
Label	0	0

Table 3: All CIC-IDS2017 Features

Approach	Scenario	F1-Score	roc-auc	pr-ap	TP	TN	FP	FN	TPR	TNR	FPR	FNR	ACC	thr	test time sec
RF	Base	0.9942	1.0000	0.9999	83454	339155	1569	4	1.0000	0.9954	0.0046	0.0000	0.9963	0.0095	1.1781
IF	Base	0.7069	0.7296	0.4739	36584	317993	22731	46874	0.4384	0.9333	0.0667	0.5616	0.8359	0.5034	1.6187
ET-SSL	Base	0.8611	0.9477	0.8214	69128	316119	24605	14330	0.8283	0.9278	0.0722	0.1717	0.9082	0.0008	0.0000
RF	Zero-Day	0.9931	0.9989	0.9982	83185	339137	1587	273	0.9967	0.9953	0.0047	0.0033	0.9956	0.0033	1.2424
IF	Zero-Day	0.7069	0.7296	0.4739	36584	317993	22731	46874	0.4384	0.9333	0.0667	0.5616	0.8359	0.5034	1.6344
ET-SSL	Zero-Day	0.8512	0.9248	0.6530	67588	314926	25798	15870	0.8098	0.9243	0.0757	0.1902	0.9018	0.0002	0.0000

Table 4: All Findings

Declaration

Ich erkläre ehrenwörtlich:

dass ich die vorliegende Hausarbeit in allen Teilen selbstständig angefertigt und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel benutzt habe, und dass die Arbeit in gleicher oder ähnlicher Form in noch keiner anderen Prüfung vorgelegen hat. Sämtliche wörtlichen oder sinngemäßen Übernahmen und Zitate, sowie alle Abschnitte, die mithilfe von KI-basierten Tools entworfen, verfasst und/oder bearbeitet wurden, sind kenntlich gemacht und nachgewiesen. Im Anhang meiner Arbeit habe ich sämtliche KI-basierte Hilfsmittel angegeben. Diese sind mit Produktnamen und formulierten Eingaben (Prompts) in einem KI-Verzeichnis ausgewiesen.

Ich bin mir bewusst, dass die Verwendung von Texten oder anderen Inhalten und Produkten, die durch KI-basierte Tools generiert wurden, keine Garantie für deren Qualität darstellt. Ich verantworte die Übernahme jeglicher von mir verwendeter maschinell generierter Passagen vollumfänglich selbst und trage die Verantwortung für eventuell durch die KI generierte fehlerhafte oder verzerrte Inhalte, fehlerhafte Referenzen, Verstöße gegen das Datenschutz- und Urheberrecht oder Plagiate.

Berlin, den 26. August 2025

A handwritten signature in black ink, appearing to read 'C. Schuster', with a stylized, cursive script.

Cedric Schuster