electronics

MDPI

Article

Unsupervised Anomaly Detection and Explanation in Network Traffic with Transformers

André Kummerow [1,*], Esrom Abrha [1], Markus Eisenbach [2] and Dennis Rösch [1]

[1] Cognitive Energy Systems, Fraunhofer IOSB, IOSB-AST, 98693 Ilmenau, Germany
[2] Neuroinformatics and Cognitive Robotics Lab, Ilmenau University of Technology, 98693 Ilmenau, Germany
* Correspondence: andre.kummerow@iosb-ast.fraunhofer.de

Abstract: Deep learning-based autoencoders represent a promising technology for use in network-based attack detection systems. They offer significant benefits in managing unknown network traces or novel attack signatures. Specifically, in the context of critical infrastructures, such as power supply systems, AI-based intrusion detection systems must meet stringent requirements concerning model accuracy and trustworthiness. For the intrusion response, the activation of suitable countermeasures can greatly benefit from additional transparency information (e.g., attack causes). Transformers represent the state of the art for learning from sequential data and provide important model insights through the widespread use of attention mechanisms. This paper introduces a two-stage transformer-based autoencoder for learning meaningful information from network traffic at the packet and sequence level. Based on this, we present a sequential attention weight perturbation method to explain benign and malicious network packets. We evaluate our method against benchmark models and expert-based explanations using the CIC-IDS-2017 benchmark dataset. The results show promising results in terms of detecting and explaining FTP and SSH brute-force attacks, highly outperforming the results of the benchmark model.

Keywords: explainable artificial intelligence; network intrusion detection; intrusion response; anomaly detection; transformers

Citation: Kummerow, A.; Abrha, E.; Eisenbach, M.; Rösch, D. Unsupervised Anomaly Detection and Explanation in Network Traffic with Transformers. *Electronics* **2024**, *13*, 4570. https://doi.org/10.3390/electronics13224570

Academic Editors: Mustafa Abdallah and Xiao Luo

Received: 14 October 2024
Revised: 15 November 2024
Accepted: 19 November 2024
Published: 20 November 2024

1. Introduction

1.1. Motivation

Network intrusion detection systems (NIDSs) play a vital role in detecting cyber-attacks and are an important component in today's security operation centers. Machine learning techniques, especially deep learning models, can learn the behavior in network traces from large network infrastructures. This enables the detection of complex cyber-attacks from flow-based or packet-based network features. Transformers represent the state of the art in the processing of sequential data and are applied in various application domains [1–4]. Thus, they are a promising approach with which to process and analyze the heterogenous information in long network packet sequences.

The limited availability of attack examples as well as the constant emergence of new attack patterns [5–8] requires unsupervised model training to efficiently distinguish between benign and malicious network packet sequences. Additional transparency information about the model's decisions (e.g., attack causes) can support subsequent decision-making tasks for intrusion prevention or forensic analysis. This increases the reliability and trustworthiness of the model's decisions and improves the further deployment of data-driven NIDS applications in the industry.

1.2. Related Work

The application of deep learning models to detect network anomalies or to classify attack traces has been investigated for many years [7–10]. Most of the research focuses on

solutions that detect cyber-attacks in a classification setting (e.g., using transformers [11], CNNs [12], or CNN-LSTMs [13]). Anomaly-based cyber-attack detection is significantly more challenging since it has no prior knowledge of attack examples in the training phase. Classical machine learning approaches apply statistical analysis, outlier detection models, or clustering methods to preprocessed and attributed network traffic records. In [14], the authors introduce an empirical anomaly score learning method with regularized self-representations on network flows using the Bot-IoT [15] and USNW-NB15 [16] datasets. An intrusion detection and prevention framework is presented in [17] that uses deep autoencoders, OC-SVM, and DBSCAN clustering on network flow features from the CSE-CIC-IDS2018 [18] dataset. A multi-dimensional anomaly detection method that uses Poisson canonical polyadic decomposition (CPD) is evaluated in [19] via its application to host authentication events, network flow records, and banking transaction logs. The authors of [20] combine an ensemble clustering approach with a neural network classifier to detect cyber-attacks in the NSL-KDD [21] and TON-IoT [22] datasets. Lastly, [23] tested six classical anomaly detection algorithms (e.g., Isolation Forest) on the NSL-KDD and ISCX-2012 [24] datasets.

Deep learning-based approaches use traditional multi-layer perceptron (MLP)-based autoencoders (AEs) on the statistical features of preprocessed and benign network traffic records. This is performed in [25] to detect probing or denial-of-service attacks in the NSL-KDD dataset and employed in [26] to detect botnet attacks in IoT networks. An ensemble of AEs is presented in [27] to detect flooding or man-in-the-middle attacks on video surveillance networks. Similar work is performed in [28] to detect cyber-attacks on an experimental smart home network. Additionally, extensive hyperparameter research related to MLP-based AEs is performed in [29] on various benchmark datasets (e.g., NSL-KDD or IoTID20 [30]).

Similar to our approach, in [31], a transformer model with which to forecast network packet sequences is presented. This uses the vanilla encoder–decoder architecture introduced in [1]. They process categorical, binary, and numerical features from network packet sequences to detect flooding and scanning attacks in ICS networks. As a downside, this approach does not incorporate explainability aspects and compresses the individual packet information through linear transformation, making it difficult to identify its relevance to benign or malicious network behavior. Our approach processes discrete and continuous network packet information through individual transformations and calculates a common feature representation with an interpretable attention mechanism.

Explainable artificial intelligence (XAI) is a well-studied topic, especially regarding deep learning models [32–35]. For the transparent detection of cyber-attacks, only a little research has been carried out. This includes an approach specific to variational autoencoders, outlined in [36], and the anomaly contribution explainer (ACE) outlined in [37], which assumes a vectorized input and a scalar output (anomaly score). The field of transparent anomaly detection and anomaly explanation [38] offers alternatives with which to compute transparency information after detecting cyber-attacks. In [39], layer-wise relevance propagation is used to explain anomalies detected by a binary MLP-based classifier. The work in [40] extends the well-known Shapley additive explanations (SHAP) approach to explain anomalies from the reconstruction errors of deep autoencoders, but requires vectorized model inputs and outputs, as in the case of ACE. The DAEMON approach outlined in [41] uses adversarial training for a CNN autoencoder with known prior distribution of the latent space. In [42], proximate counterfactuals are generated to explain anomalies detected by deep autoencoders. These previously used explanation techniques are not appropriate for our study as they have excessively strict limitations in terms of the anomaly detection model. Attention-based explanation methods are more related to our modeling approach. Here, the ATON (attention-guided triplet deviation network) [43] calculates anomaly scores with an attention coefficient vector and determines the contribution of each embedding dimension to the model decision. This approach requires a specific neural network architecture and a specific loss formulation. In contrast to that, the attention manipulation (AtMan) approach of [44] is more model-agnostic and perturbs the attention scores of a transformer model to compute their contribution to the

model's decision. We adapt their approach to implement a sequential explanation procedure for our combined autoencoder and detection model.

1.3. Paper's Contribution and Organization

To detect and explain malicious network packets, we propose using a transformer-based autoencoder to learn the benign network behavior in an unsupervised manner by efficiently reconstructing the discrete and continuous packet information in network traffic sequences. In analogy to natural language processing, we treat the sequential network packet information like words in a sentence using vector representations. The inherent use of multiple attention mechanisms and their analysis allows for the provision of additional transparency information at the sequence and packet level. The contribution of this paper can be summarized as follows:

- We outline a transformer-based autoencoder with feature- and self-attention mechanisms to reconstruct discrete and continuous information from network packet sequences;
- We perform unsupervised detection of malicious network packets with transformer-based autoencoders in network packet sequences;
- We offer sequential attention perturbation method for explaining the detection of benign and malicious network packets;
- We perform the evaluation of the explanation results by comparison with benchmark methods and expert-based true explanations.

The paper is organized as follows. Section 2 explains the model structure and training objective of the proposed transformer-based network traffic autoencoder. Based on this, the cyber-attack detection and explanation methods are given in Section 3. A comprehensive description of the experimental results using the CIC-IDS-2017 [45] benchmark dataset is provided in Section 4. Section 5 summarizes the research findings and gives an overview about future work.

2. T-NAE: Transformer-Based Network Traffic Autoencoder

The general structure of the proposed transformer-based network traffic autoencoder (*T-NAE*) is given in Figures 1 and 2 and is based on previous work in [46,47]. Section 2.1 describes the input data and the preprocessing steps. A detailed description of the different encoder and decoder components of the model is given in Section 2.2. Section 2.3 explains the training objective of the model.

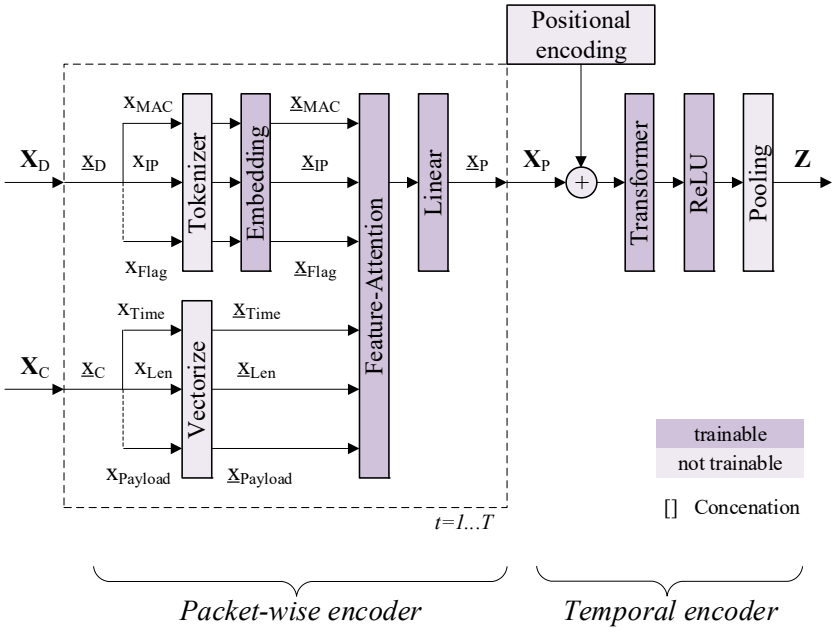

Figure 1. The encoder structure of the transformer-based network traffic autoencoder.

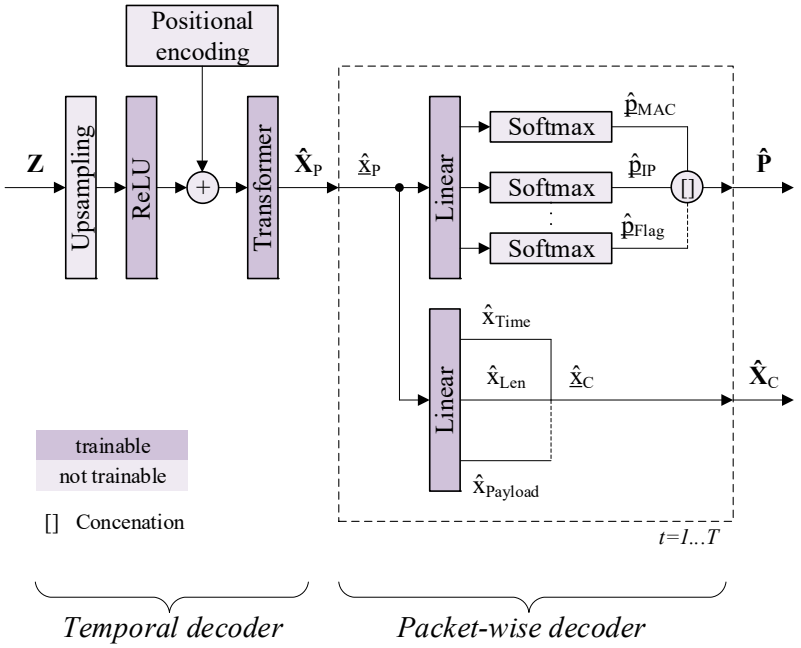

Figure 2. The decoder structure of the transformer-based network traffic autoencoder.

2.1. Preprocessing of Discrete and Continuous Network Packet Information

The efficient processing of network packet information in the T-NAE model requires some transformation steps. The discrete packet information (see Table 1) is tokenized using a fixed vocabulary with V entries.

Table 1. Overview of discrete input features.

Input Feature	Encoding Type
Packet structure (trimmed)	[1...N]
MAC source address MAC destination address	[1...N]
IP source address IP destination address	0: no address 1: equals previous source address 2: equals previous destination address 3: new address
TCP source port TCP destination port	0: no port 1: system port 2: register port 3: dynamic port
TCP flag	[1...N]

To limit the size of the vocabulary and to improve the model's generalization capability, the following preprocessing steps are applied:

- We reduce the extracted packet structures to a combination of common network layers (e.g., Ethernet, IP, TCP, HTTP);
- We transform the IP addresses and the TCP ports into a 4-tuplet (see Table 1).

This allows the model to handle the dynamic assignment of IP addresses (e.g., via DHCP services) as well as the dynamic usage of TCP ports from the network clients. Only the MAC address and TCP flag information are taken as raw inputs. Missing packet information (e.g., IP address of UDP packets) is treated as a special token. The continuous packet information is listed in Table 2. The packet time difference is extracted from the ethernet layer timestamps of the current and previous network packets. The raw inputs are normalized before being passed into the neural network, whereas missing packet

information is treated as zero. In total, the T-NAE model processes D discrete and C continuous input features from each network packet.

Table 2. Overview of continuous input features.

Input Feature (Unit)	Encoding Type
Packet time difference (ms)	Numeric
Packet length (bytes)	
IP checksum	
IP length (bytes)	
TCP checksum	
TCP payload size (bytes)	
TCP stream number	
TCP time delay (ms)	
TCP window size	

2.2. Two-Stage Encoder and Decoder Structure

From the encoder–decoder architecture shown in Figures 1 and 2, we distinguish between a packet level (packet-wise encoder/decoder) and a sequence level (temporal encoder/decoder) by learning from network packets. At the packet level, the discrete and continuous inputs are transformed so that there is a single feature vector per network packet. At the sequence level, we compute a compact latent representation based on the learned temporal dependencies existing between the packet feature vectors.

The encoder learns a latent representation $\boldsymbol{Z}$ from a given sequence of T network packets, including the discrete information $\boldsymbol{X}_{\mathrm{D}}$ and the continuous information $\boldsymbol{X}_{\mathrm{C}}$, such that

$$\boldsymbol{X}_{\mathrm{D}} = \left[\underline{x}_{\mathrm{D}}^{t=1} \ldots \underline{x}_{\mathrm{D}}^{t=\mathrm{T}}\right] \text{with } \boldsymbol{X}_{\mathrm{D}} \in \mathbb{R}^{\mathrm{T}\times\mathrm{D}} \tag{1}$$

$$\boldsymbol{X}_{\mathrm{C}} = \left[\underline{x}_{\mathrm{C}}^{t=1} \ldots \underline{x}_{\mathrm{C}}^{t=\mathrm{T}}\right] \text{with } \boldsymbol{X}_{\mathrm{C}} \in \mathbb{R}^{\mathrm{T}\times\mathrm{C}} \tag{2}$$

The *packet-wise encoder* $f_{\mathrm{Enc,P}}(\cdot)$ includes a feature-attention layer (derived from [48]) and a linear layer with Q hidden units (see Figure 1). The computation of the feature vector $\underline{x}_{\mathrm{P}} \in \mathbb{R}^{\mathrm{Q}}$ for each network packet in the sequence is given by

$$\underline{x}_{\mathrm{P}} = f_{\mathrm{Enc,P}}([\underline{x}_{\mathrm{D}}, \underline{x}_{\mathrm{C}}], \theta_{\mathrm{Enc,P}}) \tag{3}$$

Hence, the weights and biases $\theta_{\mathrm{Enc,P}}$ are shared across all network packets. For this, each piece of discrete packet information is mapped via vector embedding using the vocabulary with V entries. The feature-attention layer combines discrete and continuous packet information vectors as a weighted sum into a single vector representation $\underline{x}_{\mathrm{A}}$ with

$$\underline{x}_{\mathrm{A}} = \sum_{i=1}^{i=\mathrm{D}+\mathrm{C}} \alpha_i \underline{x}_i \tag{4}$$

Therefore, the continuous inputs are vectorized to match the dimensions of the discrete inputs (size of the vocabulary V). The attention weights α_i are computed for each piece of discrete and continuous packet information $\underline{x}_i$ as follows:

$$s_i = \tanh(\boldsymbol{W}_S \underline{x}_i + \underline{b}_S) \tag{5}$$

$$\alpha_i = \frac{exp(s_i)}{\sum_i exp(s_i)} \tag{6}$$

More detailed information is provided in [48]. The *temporal encoder* $f_{\mathrm{Enc,T}}(\cdot)$ includes a transformer block with a self-attention mechanism followed by a linear layer with P hidden units and a 1D average pooling operation. A more detailed overview of the transformer block used is given in Figure 3.

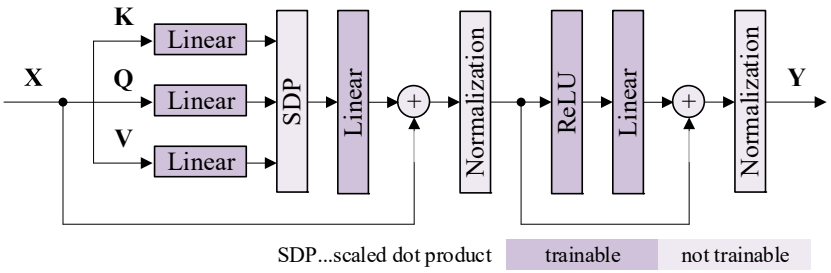

Figure 3. The structure of a multi-head self-attention block (K: keys; Q: queries; V: values).

The temporal encoder computes the latent representation $\boldsymbol{Z} \in \mathbb{R}^{\mathrm{M}\times\mathrm{P}}$ from the sequence of packet feature vectors $\boldsymbol{X}_{\mathrm{P}}$ such that

$$\boldsymbol{Z} = f_{\mathrm{Enc,T}}(\boldsymbol{X}_{\mathrm{P}}, \theta_{\mathrm{Enc,T}}) \tag{7}$$

The 1D average pooling (see Figure 1) further enhances the compression of the latent space by reducing the sequence length to M steps.

The decoder reconstructs the discrete information $\hat{\boldsymbol{P}}$ and the continuous information $\hat{\boldsymbol{X}}_{\mathrm{C}}$ using the input from $\boldsymbol{Z}$. Here, the *temporal decoder* $f_{\mathrm{Dec,T}}(\cdot)$ uses an upsampling operation, followed by a dense layer and a transformer block, to compute the reconstructed sequence of packet feature vectors $\hat{\boldsymbol{X}}_{\mathrm{P}} \in \mathbb{R}^{\mathrm{T}\times\mathrm{Q}}$ as follows:

$$\hat{\boldsymbol{X}}_{\mathrm{P}} = f_{\mathrm{Dec,T}}(\boldsymbol{Z}, \theta_{\mathrm{Dec,T}}) \tag{8}$$

The *packet-wise decoder* $f_{\mathrm{Dec,P}}(\cdot)$ includes a dense and softmax layer with which to compute a probability vector for each piece of discrete packet information (e.g., $\underline{\hat{p}}_{\mathrm{MAC,src}} \in \mathbb{R}^{\mathrm{V}}$) from a given reconstructed packet feature vector $\underline{\hat{x}}_{\mathrm{P}}$. This results in a probability matrix $\hat{\boldsymbol{P}}_{\mathrm{D}} \in \mathbb{R}^{\mathrm{D}\times\mathrm{V}}$ for a single network packet. We use another dense layer to compute the reconstructed continuous packet information $\underline{\hat{x}}_{\mathrm{C}}$ from the same feature vector $\underline{\hat{x}}_{\mathrm{P}}$ such that

$$\left[\hat{\boldsymbol{P}}_{\mathrm{D}}, \underline{\hat{x}}_{\mathrm{C}}\right] = f_{\mathrm{Dec,P}}(\underline{\hat{x}}_{\mathrm{P}}, \theta_{\mathrm{Dec,P}}) \tag{9}$$

As in case of the packet-wise encoder, we apply the packet-wise decoder for each network packet within the sequence with shared weights $\theta_{\mathrm{Dec,P}}$. This results in a probability tensor $\hat{\boldsymbol{P}} \in \mathbb{R}^{\mathrm{T}\times\mathrm{D}\times\mathrm{V}}$ for the reconstructed discrete packet information with

$$\hat{\boldsymbol{P}} = \left[\hat{\boldsymbol{P}}_{\mathrm{D}}^{t=1} \ldots \hat{\boldsymbol{P}}_{\mathrm{D}}^{t=\mathrm{T}}\right] \tag{10}$$

2.3. Model Training and Hyperparameter Optimization

The T-NAE model is trained by minimizing the reconstruction errors between the estimated ($\hat{p}$) and true (p) discrete packet information with a cross-entropy loss L_{CE} and by minimizing the reconstruction errors between the estimated ($\hat{x}_{\mathrm{C}}$) and true (x_{C}) continuous packet information with a mean squared error loss L_{MSE}. This results in a combined loss formulation as a weighted sum of cross-entropy and mean squared error loss over all T packets in the sequence such that

$$L_{\mathrm{CE}} = -\sum_{T}\sum_{D}\sum_{V} p \log \hat{p} \tag{11}$$

$$L_{\mathrm{MSE}} = \sum_{T}\sum_{C}(\hat{x}_{\mathrm{C}} - x_{\mathrm{C}})^2 \tag{12}$$

$$L = w_{\mathrm{CE}} L_{\mathrm{CE}} + w_{\mathrm{MSE}} L_{\mathrm{MSE}} \tag{13}$$

The optimal hyperparameters of the model are estimated by Bayesian optimization using Gaussian processes with 100 trials. The total loss L is used as an objective function. We choose early stopping as a regularization technique.

3. Cyber-Attack Detection and Explanation

3.1. Histogram-Based Threshold Computation

To differentiate between normal and abnormal network packets, we derive the threshold value τ as the maximum allowed reconstruction error from the training results. In the case of discrete values, the detection result $\hat{y}_{\mathrm{D}}$ is calculated from the categorical accuracy η_{ACC} between the estimated $(\underline{\hat{p}})$ and true $(\underline{p})$ packet information probability vectors as follows:

$$\hat{y}_{\mathrm{D}} = \begin{cases} \text{Attack}: & \eta_{\mathrm{ACC}}\left(\underline{\hat{p}}, \underline{p}\right) < \tau_{\mathrm{D}} \\ \text{Normal}: & \text{else} \end{cases} \tag{14}$$

A threshold value τ_{D} is estimated as the upper bound for each of the $D = 7$ discrete input features using a histogram analysis of the reconstruction error distribution (see Appendix A). For continuous values, the detection result $\hat{y}_{\mathrm{C}}$ is calculated from the mean squared error η_{MSE} between the estimated $(\underline{\hat{x}})$ and true $(\underline{x})$ packet information such that

$$\hat{y}_{\mathrm{C}} = \begin{cases} \text{Attack}: & \eta_{\mathrm{MSE}}(\underline{\hat{x}}, \underline{x}) > \tau_{\mathrm{C}} \\ \text{Normal}: & \text{else} \end{cases} \tag{15}$$

where a threshold value of τ_{C} is estimated as the lower bound for each of the $C = 9$ continuous inputs using the error histograms. The final detection result $\hat{y}$ for a single network packet follows an OR combination of the detection results over all discrete and continuous information, such that

$$\hat{y} = \left\{ \hat{y}_{\mathrm{D},1} \bigvee \ldots \bigvee \hat{y}_{\mathrm{D},7} \bigvee \hat{y}_{\mathrm{C},1} \bigvee \ldots \bigvee \hat{y}_{\mathrm{C},9} \right\} \tag{16}$$

3.2. Explanation via Perturbation of Attention Weights

Our explanation method builds upon the attention perturbation approach described in [44]. Here, the attention weights are manipulated to assess their influence on the model's results by comparing the model's loss before and after the perturbation within a positional influence function. Our approach uses a manipulation signal $a = 1 - \mathrm{f}_a$ with the following features:

- A positive manipulation factor $1 > \mathrm{f}_a > 0$ (*suppression*) to decrease the influence of a specific attention weight;
- A negative manipulation factor $\mathrm{f}_a < 0$ (*amplification*) to increase the influence of a specific attention weight.

We adapted this approach to account for the specific architecture of our T-NAE model (see Section 2) and modified the calculation of the influence values by incorporating the anomaly detection model from Section 3.1.

The explanation model is given in Figure 4. We manipulate the feature-attention weights by multiplying the attention scores (see Equation (5) in Section 2.2) with a manipulation vector $\underline{a}_{FA} \in \mathbb{R}^{\mathrm{D+C}}$. Thus, the manipulated feature-attention weights $\underline{\alpha}^*$ are calculated by

$$\underline{\alpha}^* = \mathrm{softmax}(\underline{s} * \underline{a}_{FA}) \tag{17}$$

The self-attention mechanism in the transformer block (Section 2.2) computes attention scores, $\boldsymbol{S} \in \mathbb{R}^{\mathrm{T}\times\mathrm{T}}$, via scaled-dot products over the key $\underline{q}$ and value $\underline{k}$ representations as follows:

$$\boldsymbol{S} = [s_{i,j}] \text{ with } s_{i,j} = \underline{q}_i^T \underline{k}_j / \sqrt{d} \text{ and } i, j \in [1 \ldots \mathrm{T}] \tag{18}$$

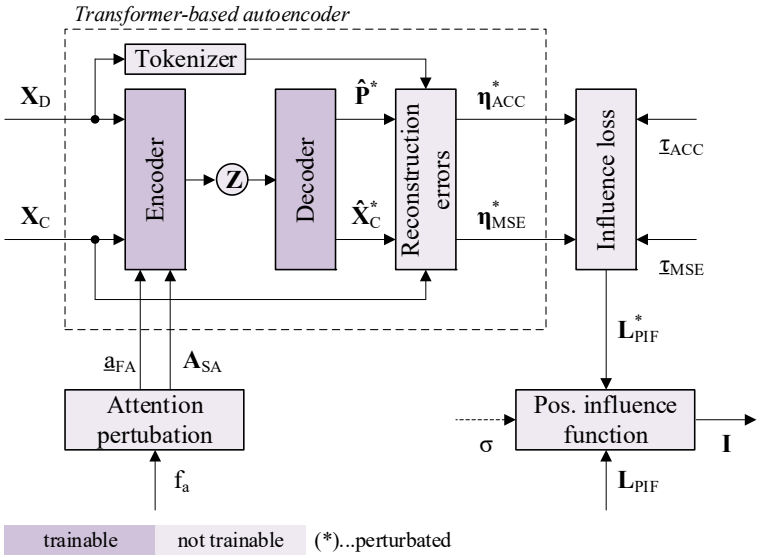

Figure 4. Explanation model architecture with attention perturbation.

Thus, we compute the manipulated self-attention weights $\boldsymbol{\alpha}^*$ by multiplying the attention scores with a manipulation matrix $\boldsymbol{A}_{SA} \in \mathbb{R}^{\mathrm{T}\times\mathrm{T}}$:

$$\boldsymbol{\alpha}^* = \mathrm{softmax}(\boldsymbol{S} * \boldsymbol{A}_{SA}) \tag{19}$$

It should be noted that a manipulation of the feature or self-attention scores leads to a change in both model outputs (discrete and continuous reconstructions). This has to be considered when calculating the influence values in the following section.

We compute the loss for the positional influence function as the distance between the reconstruction error, averaged over the timesteps of the sample, and the threshold value with $L_{\mathrm{PIF,D}} = \eta_{\mathrm{ACC}} - \tau_{\mathrm{D}}$ and $L_{\mathrm{PIF,C}} = \eta_{\mathrm{MSE}} - \tau_{\mathrm{C}}$. The influence values I are calculated from the difference in the losses attained with perturbation L_{PIF}^* and without perturbation L_{PIF}. For consistency, we declare that positive influence values $I > 0$ contribute to the detection of normal samples and that negative influence values $I < 0$ contribute to the detection of attack samples. This gives the following calculations of the influence values for discrete and continuous reconstructions:

$$I_D = f\left(L_{\mathrm{PIF,D}}^*\right) - f\left(L_{\mathrm{PIF,D}}\right) \tag{20}$$

$$I_C = f\left(L_{\mathrm{PIF,C}}\right) - f\left(L_{\mathrm{PIF,C}}^*\right) \tag{21}$$

To improve the interpretation of the explanation results, we bound the influence values into the range of $[-1;1]$ by converting the loss values with a Sigmoid function as follows:

$$f_{\mathrm{Sig}}(L_{\mathrm{PIF}}) = 1/\left(1 + \exp\left(-\frac{L_{\mathrm{PIF}}}{\sigma}\right)\right) \tag{22}$$

The bandwidth parameter σ gives additional control over the explanation results by assigning high influence values to loss values that are close to the threshold regime. This will focus the explanation results on the input signals that contribute most to the decisions of the detection model. An exemplary illustration is given in Figure 5. To accommodate and aggregate the influence values related to the feature-attention and self-attention scores, we follow the sequential procedure shown in Figure 6. The corresponding steps I to IV of the procedure are explained in the following material.

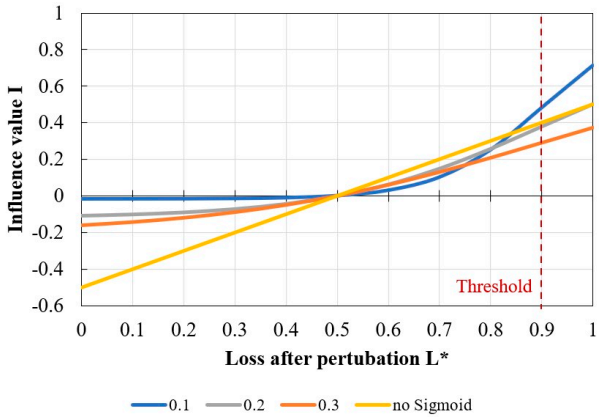

Figure 5. Influence values for exemplary positional losses $L^*_{PIF,D}$ using a Sigmoid function with different bandwidth parameters ($L_{PIF,D} = 0.5$).

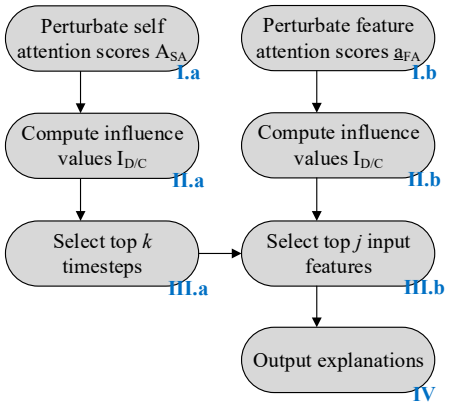

Figure 6. Explanation procedure with perturbation of attention weights.

Within each perturbation step (*I.a* and *I.b*), we perform multiple manipulation runs. In each run, we calculate influence values by applying a manipulation signal to only one specific attention score, whereas the rest of the attention scores remain unchanged with $a = 0$. First, we perturb the self-attention scores and identify the top k influential network packets (time steps) within the given sequence (*I.a* to *III.a*). This is performed by summing up the influence values over all K manipulation runs with

$$\underset{k}{\operatorname{argmax}} \sum_{i=1}^{i=\mathrm{K}} I^i_{D/C} \tag{23}$$

We perform this separately for all positive influence values ($I_{D/C} > 0$) and negative influence values ($I_{D/C} < 0$), resulting in $2k$ most influential network packets or time steps being identified (*III.a*). In parallel, we perturb the feature-attention scores and compute the corresponding influence values for all network packets and their input features (*I.b* and *II.b*). Finally, we select the top j most influential input features using the procedure from above for the top k most influential network packets (*III.b*). This results in a $2k \times 2j$ matrix of influence values for the discrete $\boldsymbol{I}_D$ and continuous $\boldsymbol{I}_C$ reconstructions (*IV*).

4. Results and Discussion

4.1. CIC-IDS-2017 Dataset and Explanation Evaluation

For our evaluation, we use a network traffic excerpt from the CIC-IDS-2017 dataset. The training set contains only benign network traces, whereas the test set contains benign and malicious network traces from FTP and SSH brute-force attacks. Here, we assume that

the benign network behavior does not change significantly between the training and test dataset. An overview of the two datasets is given in Table 3.

Table 3. Overview training and test datasets.

Dataset	Characteristics
Training	Normal packets: 750,000 Attack packets: 0 Attack ratio: 0%
Test	Normal packets: 752,600 Attack packets: 97,400 Attack ratio: 13%

We evaluate the detection performance of our T-NAE model (see Section 3.1) by calculating the F_1 score, the binary accuracy (BA), and the false positive rate (FPR) using the test dataset.

We compare our explanation results with those obtained when the integrated gradient (IG) method [49] is applied to our T-NAE model (see Section 2). The comparison of the explanation results is performed for selected samples from the test dataset, including

- Ten normal samples indicating web browsing and FTP file transfer activities;
- Eighteen attack samples indicating FTP and SSH brute-force attacks.

For these test samples, we estimate true explanations, using expert knowledge to better assess the explanation results. For this, we estimate the true influence values according to the following criteria:

- For normal samples, we assign influence values of $I = 1$ for packets containing TCP, HTTP, or FTP protocol layers and specific MAC addresses;
- For attack samples, we assign influence values of $I = -1$ for packets containing TCP, SSH, or FTP protocol layers as well as the attacker's MAC address.

The expert-based explanations are limited to the protocol type and MAC address information only. Since we do not know the relevance of the remaining packet information, we assign it influence values of zero with $I = 0$. To compare the estimated and true influence values, we compute the F_β score between the predicted and true explanation values, which allows for additional control of the influence of the precision and the recall. For our evaluation, we choose a low β value ($\beta = 0.15$) to account for the high number of zero entries in the true explanations.

4.2. Reconstruction Accuracies and Detection Results

Figure 7 shows the performance of a T-NAE model over roughly 600 training epochs. The left chart shows the training and validation loss and the right chart shows the reconstruction results for discrete (accuracy) and continuous (mean squared error) model outputs. Good model convergence can be seen, with small differences between the training and validation loss curves. During the training, the accuracy values for the discrete reconstructions increase, whereas the squared error values for the continuous reconstructions decrease. Figure 8 shows the average reconstruction results over the different discrete and continuous model outputs for the normal and attack samples of the test dataset. In the case of the discrete features (left panel), the accuracy results of the attack samples are below the accuracy results of the normal samples. In particular, this holds for the MAC source and destination addresses, indicating a good separability between the normal and attack network packets. This is not the case for the continuous features (right panel). Here, only the features with high MSE values show good separability between normal and attack samples, whereas the other features show very low and similar reconstruction results. In the experiments, we observe the best detection results using the reconstruction errors of the discrete model outputs. This is considered in the evaluation of the explanation results in Section 4.3.

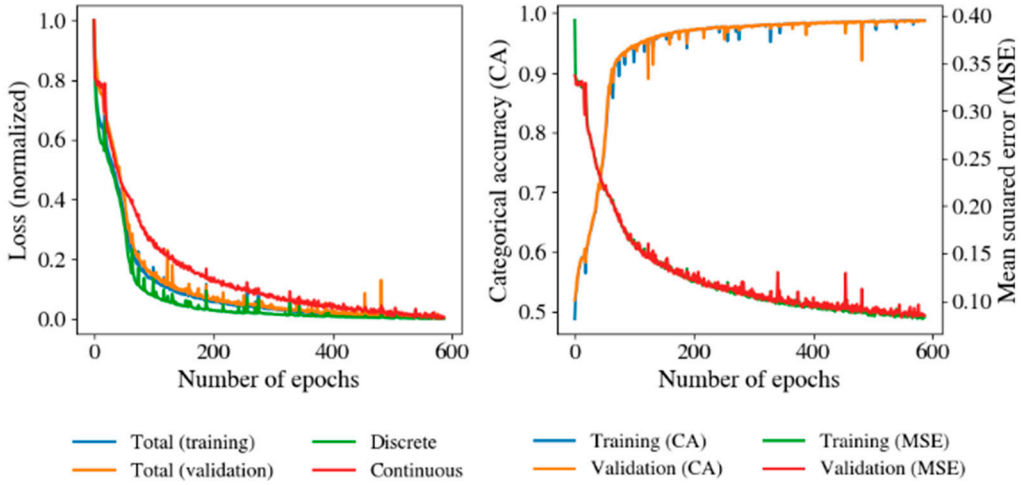

Figure 7. Training results of a T-NAE model (left: loss values; right: reconstruction values).

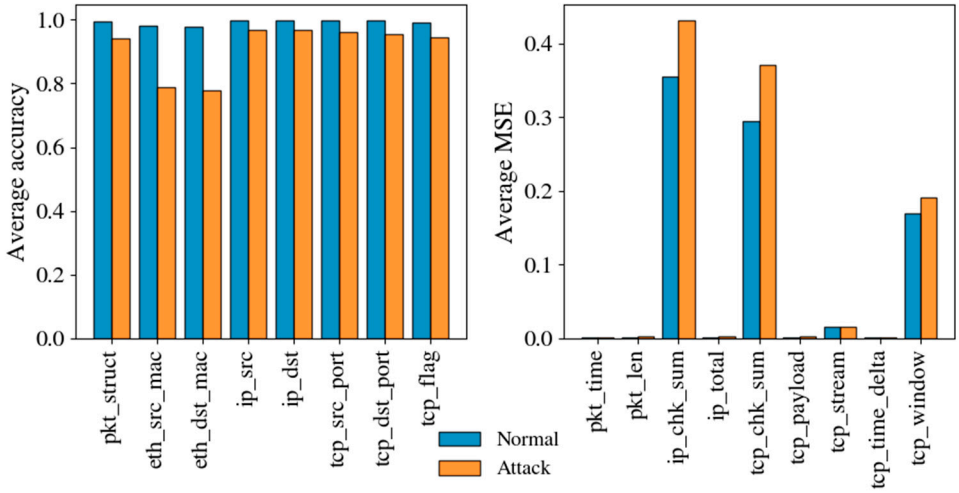

Figure 8. Reconstruction results of using a T-NAE model on the test dataset.

The detection performance is given in Table 4 for T-NAE models with different pool sizes of the encoder (see Section 2.2). For all models, the hyperparameters are tuned as described in Section 2.3. The best T-NAE model gives an F_1 score of about 95% with a false positive rate of about 4.5%. This model is used for the subsequent explanation experiments in Section 4.3. The corresponding parameters of the T-NAE model and the detection model are given in Appendix B.

Table 4. Detection results of applying different T-NAE models to the test dataset (values in %).

Pool Size	F_1	BA	FPR
25	95.36	95.47	4.53
50	91.59	92.17	7.83
100	74.02	79.42	20.58

4.3. Explanation of Benign and Malicious Network Packet Sequences

Figures 9 and 10 show the explanation results for two exemplary samples. Each sample consists of a sequence of $T = 100$ network packets. The figures show the influence values at the feature level (top left and right panels) and at the sequence level (bottom panel). As already mentioned in Section 4.2, the influence values shown result from the

discrete reconstructions, such that $I = I_D$. In Figure 9, most of the influence values are positive, indicating a benign network behavior. The influence values are quite small with $|I_D| < 0.06$. The TCP flag and MAC address features contribute the most to the model's predictions, whereas the continuous features show no importance or relevance. Negative influence values can only be seen for some network packets, but the positive influence values predominate along the entire sequence. In Figure 10, the number and size of the negative influence values are significantly larger, indicating malicious network behavior. In particular, the MAC and IP address features exhibit a strong impact on the detection results.

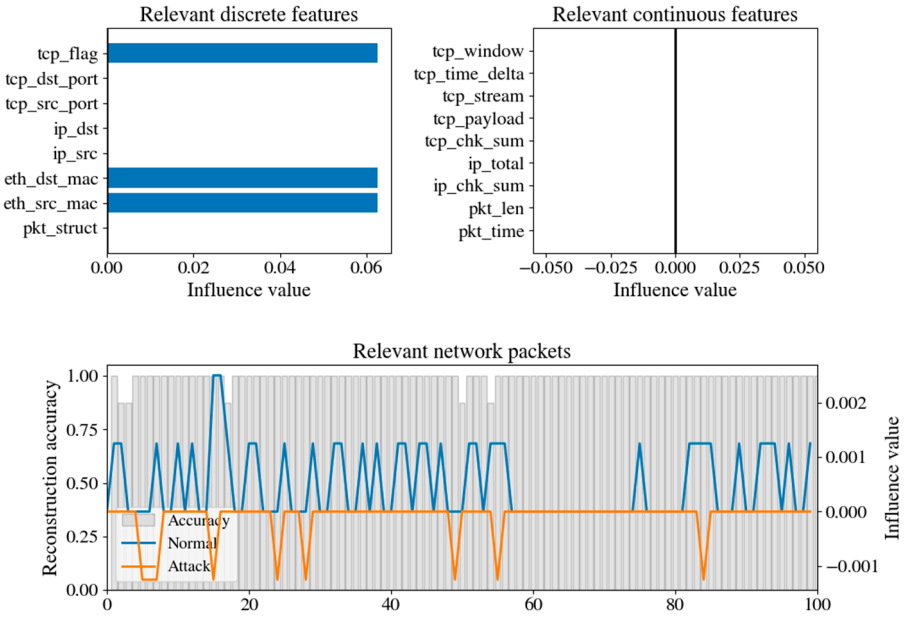

Figure 9. Influence values (top left: discrete inputs; top right: continuous inputs; bottom: packet sequence) for an exemplary benign network packet sequence.

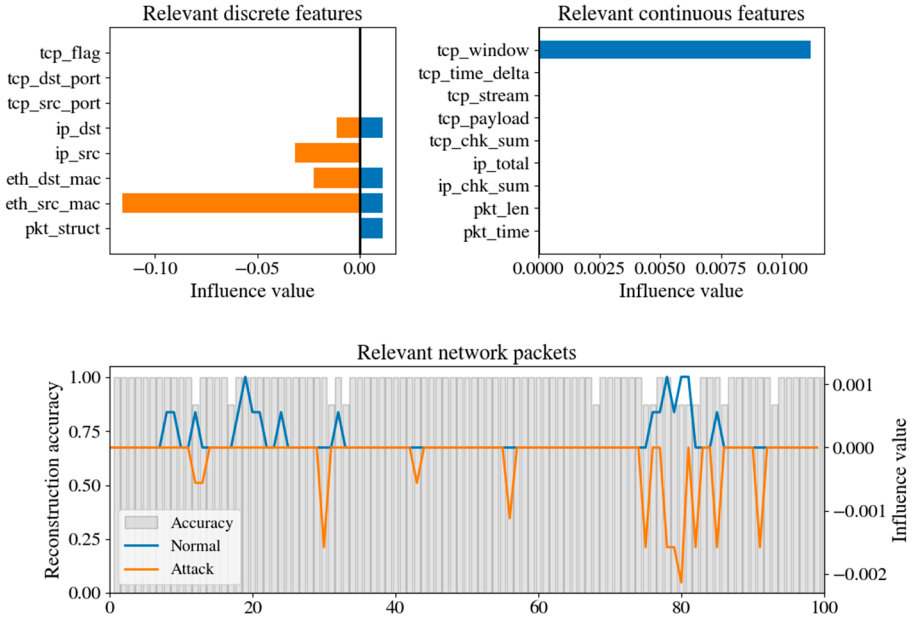

Figure 10. Influence values (top left: discrete inputs; top right: continuous inputs; bottom: packet sequence) for an exemplary malicious network packet sequence.

As in the previous case, the continuous features show very low influence values or influence values of zero. Negative influence values can be clearly observed for the network

packets at specific time steps in the sequence. In particular, between $T = 70$ and $T = 90$, there are relatively high positive and high negative influence values for the same network packets, with a slight predominance of the negative influence values. As in the previous case, the influence values are quite small, with $|I_D| < 0.1$.

As introduced in Section 4.1, we compare the influence values for the expert-labeled normal and attack samples with the IG method's results using the F_β score. Within the experiments, we vary the parameters of the explanation model (see Section 3.2), including the manipulation factor f_a, the maximum number of most influential network packets k, the maximum number of most influential input features j, and the bandwidth of the positional influence function σ. As also stated in [44], we suppress the attention scores by applying positive manipulation factors throughout the perturbation experiments, which gives us better explanation results compared to those obtains from amplification.

For a better evaluation of the explanation results, we additionally group the normal and attack samples according to their average detection accuracy results. Tables 5 and 6 show the explanation results for the best hyperparameters of the explanation model. The results for other configurations of the explanation model, including negative manipulation factors, are given in Appendix C. In the case of normal samples, the perturbation-based explanations yield F_β scores between 22% and 24% and significantly exceed the F_β scores of the IG-based explanations (which are between 6% and 7%). Nonetheless, the F_β scores of the normal samples are quite low for both methods. Unlike the case of the attack samples, we have no side information for the normal network traffic behavior. This makes the derivation of expert-based explanations for normal samples quite hard and may lead to incomplete or erroneous ground truths of the influence values. In the case of the attack samples, the F_β scores of both methods are greatly increased. Again, the perturbation-based explanation results (F_β scores between 54% and 69%) clearly surpass the IG-based explanation results (F_β scores between 18% and 19%). On average, the explanation results for the samples with low detection accuracy are slightly better than those for the samples with high detection accuracy.

Table 5. Best explanation results for samples with high detection accuracy (with $\beta = 0.15$).

Samples	f_a	k	j	σ	F_β (Ours) [%]	F_β (IG) [%]
Normal	0.10	18	1	0.4	22.3	7.4
Attack	0.99	22	1	0.3	54.6	18.6

Table 6. Best explanation results for samples with low detection accuracy (with $\beta = 0.15$).

Samples	f_a	k	j	σ	F_β (Ours) [%]	F_β (IG) [%]
Normal	0.8	28	1	0.4	24.3	6.0
Attack	0.2	26	1	0.4	69.5	19.4

Regarding the hyperparameters of our explanation model, we observe that, in general, high values of k and low values of j lead to the highest F_β scores. Low values of j allow us to filter the most relevant input features for the explanations, which corresponds to the ground truth explanations. For attack samples with high detection accuracies, high manipulation factors up to $f_a \approx 0.995$ improve the F_β scores on average. For normal samples or samples with low detection accuracies, no clear influence of the f_a values on the F_β scores can be observed. In contrast to that, the bandwidth parameter σ exhibits a low influence on the explanation results and shows the best results in the range of 0.3 to 0.4. This is consistent with previous findings showing that the perturbation of the attention weights only slightly changes the model outputs, leading to small changes within the positional influence function and to low explanation values ($|I_D| < 0.1$).

5. Summary and Outlook

In this study, we present a sequential attention weight perturbation method to explain benign and malicious packets from network traffic traces. For this purpose, we introduce a transformer-based autoencoder to learn the basic network traffic behavior and employ a histogram-based detection model using reconstruction error thresholds. The two-stage encoder–decoder architecture of the autoencoder incorporates different attention mechanisms at the packet and sequence levels. Our explanation approach manipulates the attention scores in a sequential manner to compute the input information making the largest contribution in order to explain normal or attack behavior in network packet sequences. In extensive experiments, performed using an excerpt from the CIC-IDS-2017 benchmark dataset, our explanation method shows superior results when applied to benign and malicious network traces (including FTP and SSH brute-force attacks) compared to the integrated gradients of benchmark methods. For the evaluation, we derived expert-based explanations for the network traces, considering the MAC address and protocol type information, and compared them with the predicted explanations (influence values) by calculating F_β scores. With an average binary detection accuracy of about 95% when applied to the test dataset, our explanation method achieves an F_β score of about 70% on attack samples, compared to the value of approximately 20% achieved by the IG method.

Still, these results are highly dependent on the chosen hyperparameters of the explanation model, and a method is required to automatically determine the hyperparameters and conduct further systematic investigations. There is also a need for other transparent cyber-attack detection benchmark models, for use on large datasets, with more reliable true explanations. In particular, this accounts for the labeling of benign network traces, where it is hard to derive the relevant explanatory factors. Here, representative datasets with already labeled explanations for normal and malicious network traces are necessary in order to systematically evaluate transparent cyber-attack detection methods at scale. Nevertheless, future experiments will be conducted to evaluate our method on a greater number of attack examples with manually derived explanations. Furthermore, the computational complexity of the explanation method should be reduced by minimizing the number of necessary perturbations. This could be achieved by grouping input features with similar influence behaviors (as already suggested in [44]) or by simultaneously perturbation at the feature and sequence levels. Alternatively, the computational time could be reduced by parallelizing the perturbation operations or by quantizing the weights of the T-NAE model. This would simplify the implementation in real-world scenarios, where a high throughput of large network traces would result in large autoencoder models. The detection model (see Section 3.1) is kept quite simple and this might lead to misclassifications in dynamic and high-noise environments. A more adaptive method should be considered in further research.

Author Contributions: Conceptualization, A.K. and M.E.; methodology, A.K., E.A. and M.E.; software, A.K. and E.A.; validation, A.K., E.A. and M.E.; formal analysis, A.K. and M.E.; investigation, E.A.; resources, D.R.; data curation, A.K.; writing—original draft preparation, A.K. and E.A.; writing—review and editing, M.E. and D.R.; visualization, A.K. and E.A.; supervision, A.K. and M.E.; project administration, A.K. and D.R.; funding acquisition, D.R. All authors have read and agreed to the published version of the manuscript.

Funding: This publication is a result of the project PROTECT (03EI6054A) funded by the German Federal Ministry for Economic Affairs and Climate Action (BMWK).

Data Availability Statement: The raw data supporting the conclusions of this article will be made available by the authors on request.

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A

As introduced in Section 3.1, the detection model uses a histogram-based error analysis to derive threshold values for discrete and continuous reconstructions. The analysis

procedure is illustrated in Figure A1 under the assumption of Gaussian distributed reconstruction errors. From the absolute frequency value h, we compute the relative cumulative sum S over all frequency bins. The threshold value is set at a specific value for S, where $\alpha = 1 - S$ is a user-defined parameter (maximum cumulative sum). Thus, a value of $\alpha = 0.95$ means that 95% of the reconstruction errors are below the threshold and 5% are above the threshold. Thus, the α parameter allows us to control the acceptable fraction of outliers within the reconstruction errors.

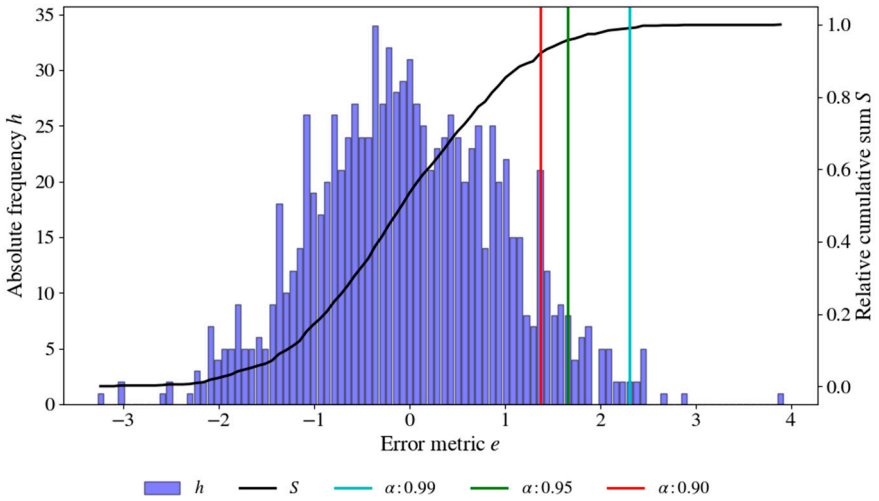

Figure A1. Histogram-based threshold calculation for exemplary Gaussian errors.

Appendix B

Table A1. Hyperparameters of the detection model.

Parameter	Value
Number of bins N	100
Maximum cumulative sum α	0.9

Table A2. Hyperparameters of the T-NAE model.

Parameter	Value
Number of embedding dimensions	50
Hidden unit linear layer (encoder)	[900, 300]
Hidden unit linear layer (decoder, discrete outputs)	[900, 200]
Hidden unit linear layer (decoder, continuous outputs)	[1000, 500]
Number of transformer blocks	1
Number of attention heads	3
Size of attention head	10
Hidden unit linear layer (transformer block)	400
Number of latent dimensions	160
Optimizer	Adam
Learning rate	0.0005
Batch size	100
Loss weight (discrete outputs)	1.0
Loss weight (continuous outputs)	1.4

Appendix C

Table A3. Explanation results for normal samples with high detection accuracy (with $\beta = 0.15$).

No.	$\mathbf{f}_a$	k	j	σ	$\eta_{\mathbf{F}-\beta}$[%]
		Suppression experiments			
1	0.1 . . . 0.99	18	1	0.4	16.9 . . . 22.3
2	0.99	18 . . . 42	1	0.4	12.5 . . . 16.2
3	0.99	18	1 . . . 4	0.4	11.7 . . . 16.1
4	0.99	18	1	0.1 . . . 0.6	16.9 . . . 17.0
		Amplification experiments			
5	−4.0 . . . −0.2	18	1	0.2	12.2 . . . 21.6
6	−1.0	18 . . . 40	1	0.4	15.5 . . . 18.2
7	−1.0	20	1 . . . 4	0.4	12.2 . . . 18.1
8	−1.0	18	1	0.1 . . . 0.6	18.7 . . . 19.2

Table A4. Explanation results for normal samples with low detection accuracy (with $\beta = 0.15$).

No.	$\mathbf{f}_a$	k	j	σ	$\eta_{\mathbf{F}-\beta}$[%]
		Suppression experiments			
1	0.05 . . . 0.995	28	1	0.4	0.0 . . . 11.7
2	0.99	18 . . . 40	1	0.4	8.4 . . . 13.3
3	0.99	20	1 . . . 4	0.4	7.6 . . . 10.9
4	0.99	28	1	0.1 . . . 0.6	11.1 . . . 12.5
		Amplification experiments			
5	−4.0 . . . −0.2	18	2	0.3	0.0 . . . 8.8
6	−1.0	18 . . . 40	2	0.4	7.6 . . . 7.9
7	−1.0	20	1. . .4	0.4	5.8 . . . 7.6
8	−1.0	18	2	0.1 . . . 0.6	7.7 . . . 7.9

Table A5. Explanation results for attack samples with high detection accuracy (with $\beta = 0.15$).

No.	$\mathbf{f}_a$	k	j	σ	$\eta_{\mathbf{F}-\beta}$[%]
		Suppression experiments			
1	0.05 . . . 0.995	22	1	0.3	40.7 . . . 52.6
2	0.99	18 . . . 42	1	0.4	51.0 . . . 54.6
3	0.99	20	1 . . . 4	0.4	28.0 . . . 51.0
4	0.99	22	1	0.1 . . . 0.6	49.4 . . . 54.6
		Amplification experiments			
5	−4.0 . . . −0.2	26	1	0.2	35.1 . . . 53.8
6	−1.0	18 . . . 40	1	0.4	46.1 . . . 46.7
7	−1.0	20	1 . . . 4	0.4	27.2 . . . 46.1
8	−1.0	26	1	0.1 . . . 0.6	44.7

Table A6. Explanation results for attack samples with low detection accuracy (with $\beta = 0.15$).

No.	$\mathbf{f}_a$	k	j	σ	$\eta_{\mathbf{F}-\beta}$
		Suppression experiments			
1	0.05 . . . 0.995	26	1	0.4	35.7 . . . 69.5
2	0.99	18 . . . 40	1	0.4	52.0 . . . 54.1
3	0.99	20	1 . . . 4	0.4	31.2 . . . 52.2
4	0.99	26	1	0.1 . . . 0.6	54.0 . . . 54.8
		Amplification experiments			
5	−4.0 . . . −0.2	18	1	0.3	32.8 . . . 76.6
6	−1.0	18 . . . 40	1	0.4	51.2 . . . 54.1
7	−1.0	20	1 . . . 4	0.4	28.3 . . . 53.7
8	−1.0	18	1	0.1 . . . 0.6	53.3 . . . 54.1

References

1. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.N.; Kaiser, Ł.; Polosukhin, I. Attention Is All You Need. 2017. Available online: http://arxiv.org/pdf/1706.03762v5 (accessed on 23 March 2023).
2. Lim, B.; Arik, S.O.; Loeff, N.; Pfister, T. Temporal Fusion Transformers for Interpretable Multi-horizon Time Series Forecasting. *Int. J. Forecast.* **2021**, *37*, 1748–1764. [CrossRef]
3. Devlin, J.; Chang, M.-W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In Proceedings of the 2019 Conference of the North, Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186.
4. Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale. *arXiv* **2020**, arXiv:2010.11929.
5. Kummerow, A.; Schäfer, K.; Gupta, P.; Nicolai, S.; Bretschneider, P. Combined Network Intrusion and Phasor Data Anomaly Detection for Secure Dynamic Control Centers. *Energies* **2022**, *15*, 3455. [CrossRef]
6. RÖsch, D.; Kummerow, A.; Ruhe, S.; Schäfer, K.; Monsalve, C.; Nicolai, S. IT-Sicherheit in digitalen Stationen: Cyber-physische Systemmodellierung, -bewertung und -analyse. *Automatisierungstechnik* **2020**, *68*, 720–737. [CrossRef]
7. Aleesa, A.M.; Zaidan, B.B.; Zaidan, A.A.; Sahar, N.M. Review of intrusion detection systems based on deep learning techniques: Coherent taxonomy, challenges, motivations, recommendations, substantial analysis and future directions. *Neural Comput. Appl.* **2019**, *32*, 9827–9858. [CrossRef]
8. Liu, H.; Lang, B. Machine Learning and Deep Learning Methods for Intrusion Detection Systems: A Survey. *Appl. Sci.* **2019**, *9*, 4396. [CrossRef]
9. Aldweesh, A.; Derhab, A.; Emam, A.Z. Deep learning approaches for anomaly-based intrusion detection systems: A survey, taxonomy, and open issues. *Knowl.-Based Syst.* **2020**, *189*, 105124. [CrossRef]
10. Lansky, J.; Ali, S.; Mohammadi, M.; Majeed, M.K.; Karim, S.H.T.; Rashidi, S.; Hosseinzadeh, M.; Rahmani, A.M. Deep Learning-Based Intrusion Detection Systems: A Systematic Review. *IEEE Access* **2021**, *9*, 101574–101599. [CrossRef]
11. Wu, Z.; Zhang, H.; Wang, P.; Sun, Z. RTIDS: A Robust Transformer-Based Approach for Intrusion Detection System. *IEEE Access* **2022**, *10*, 64375–64387. [CrossRef]
12. Lin, S.Z.; Shi, Y.; Xue, Z. Character-Level Intrusion Detection Based On Convolutional Neural Networks. In Proceedings of the 2018 International Joint Conference on Neural Networks (IJCNN), Rio de Janeiro, Brazil, 8–13 July 2018; pp. 1–8.
13. Wang, W.; Sheng, Y.; Wang, J.; Zeng, X.; Ye, X.; Huang, Y.; Zhu, M. HAST-IDS: Learning Hierarchical Spatial-Temporal Features Using Deep Neural Networks to Improve Intrusion Detection. *IEEE Access* **2018**, *6*, 1792–1806. [CrossRef]
14. Segurola-Gil, L.; Moreno-Moreno, M.; Irigoien, I.; Florez-Tapia, A.M. Unsupervised Anomaly Detection Approach for Cyberattack Identification. *Int. J. Mach. Learn. Cybern.* **2024**, *15*, 5291–5302. [CrossRef]
15. The Bot-IoT Dataset | UNSW Research. Available online: https://research.unsw.edu.au/projects/bot-iot-dataset (accessed on 7 November 2024).
16. The UNSW-NB15 Dataset | UNSW Research. Available online: https://research.unsw.edu.au/projects/unsw-nb15-dataset (accessed on 7 November 2024).
17. Kaliyaperumal, P.; Periyasamy, S.; Thirumalaisamy, M.; Balusamy, B.; Benedetto, F. A Novel Hybrid Unsupervised Learning Approach for Enhanced Cybersecurity in the IoT. *Future Internet* **2024**, *16*, 253. [CrossRef]
18. IDS 2018 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. Available online: https://www.unb.ca/cic/datasets/ids-2018.html (accessed on 7 November 2024).
19. Eren, M.E.; Moore, J.S.; Skau, E.; Moore, E.; Bhattarai, M.; Chennupati, G.; Alexandrov, B.S. General-purpose Unsupervised Cyber Anomaly Detection via Non-negative Tensor Factorization. *Digit. Threats* **2023**, *4*, 1–28. [CrossRef]
20. Ahmed, M.S.; Shah, S.M. Unsupervised Ensemble Based Deep Learning Approach for Attack Detection in IoT Network. 2022. Available online: http://arxiv.org/pdf/2207.07903 (accessed on 7 November 2024).
21. Tavallaee, M.; Bagheri, E.; Lu, W.; Ghorbani, A.A. A detailed analysis of the KDD CUP 99 data set. In Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications, CISDA 2009, Ottawa, ON, Canada, 8–10 July 2009; pp. 1–6.
22. The TON_IoT Datasets | UNSW Research. Available online: https://research.unsw.edu.au/projects/toniot-datasets (accessed on 7 November 2024).
23. Meira, J.; Andrade, R.; Praça, I.; Carneiro, J.; Bolón-Canedo, V.; Alonso-Betanzos, A.; Marreiros, G. Performance evaluation of unsupervised techniques in cyber-attack anomaly detection. *J. Ambient. Intell. Humaniz. Comput.* **2020**, *11*, 4477–4489. [CrossRef]
24. IDS 2012 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. Available online: https://www.unb.ca/cic/datasets/ids.html (accessed on 25 January 2022).
25. Aygun, R.C.; Yavuz, A.G. Network Anomaly Detection with Stochastically Improved Autoencoder Based Models. In Proceedings of the 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), New York, NY, USA, 26–28 June 2017; pp. 193–198.
26. Meidan, Y.; Bohadana, M.; Mathov, Y.; Mirsky, Y.; Shabtai, A.; Breitenbacher, D.; Elovici, Y. N-BaIoT—Network-Based Detection of IoT Botnet Attacks Using Deep Autoencoders. *IEEE Pervasive Comput.* **2018**, *17*, 12–22. [CrossRef]
27. Mirsky, Y.; Doitshman, T.; Elovici, Y.; Shabtai, A. Kitsune: An Ensemble of Autoencoders for Online Network Intrusion Detection. In Proceedings of the 2018 Network and Distributed System Security Symposium, San Diego, CA, USA, 18–21 February 2018.

28. Shahid, M.R.; Blanc, G.; Zhang, Z.; Debar, H. Anomalous Communications Detection in IoT Networks Using Sparse Autoencoders. In Proceedings of the 2019 IEEE 18th International Symposium on Network Computing and Applications (NCA), Cambridge, MA, USA, 26–28 September 2019; pp. 1–5.
29. Song, Y.; Hyun, S.; Cheong, Y.-G. Analysis of Autoencoders for Network Intrusion Detection. *Sensors* **2021**, *21*, 4294. [CrossRef] [PubMed]
30. Kang, H.; Ahn, D.H.; Lee, G.M.; Yoo, J.D.; Park, K.H.; Kim, H.K. IoT Network Intrusion Dataset. 2019. Available online: https://ocslab.hksecurity.net/Datasets/iot-network-intrusion-dataset (accessed on 17 September 2024).
31. Marino, D.L.; Wickramasinghe, C.S.; Rieger, C.; Manic, M. Self-Supervised and Interpretable Anomaly Detection Using Network Transformers. 2022. Available online: http://arxiv.org/pdf/2202.12997v1 (accessed on 15 September 2022).
32. Minh, D.; Wang, H.X.; Li, Y.F.; Nguyen, T.N. Explainable artificial intelligence: A comprehensive review. *Artif. Intell. Rev.* **2022**, *55*, 3503–3568. [CrossRef]
33. Lundberg, S.; Lee, S.-I. A Unified Approach to Interpreting Model Predictions. *arXiv* **2017**, arXiv:1705.07874.
34. Linardatos, P.; Papastefanopoulos, V.; Kotsiantis, S. Explainable AI: A Review of Machine Learning Interpretability Methods. *Entropy* **2020**, *23*, 18. [CrossRef]
35. Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Pedreschi, D.; Giannotti, F. A Survey Of Methods For Explaining Black Box Models. *ACM Comput. Surv.* **2018**, *51*, 1–42. [CrossRef]
36. Nguyen, Q.P.; Lim, K.W.; Divakaran, D.M.; Low, K.H.; Chan, M.C. GEE: A Gradient-based Explainable Variational Autoencoder for Network Anomaly Detection. In Proceedings of the 2019 IEEE Conference on Communications and Network Security (CNS), Washington, DC, USA, 10–12 June 2019.
37. Zhang, X.; Marwah, M.; Lee, I.-T.; Arlitt, M.; Goldwasser, D. ACE—An Anomaly Contribution Explainer for Cyber-Security Applications. In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 9–12 December 2019.
38. Yepmo, V.; Smits, G.; Pivert, O. Anomaly explanation: A review. *Data Knowl. Eng.* **2022**, *137*, 101946. [CrossRef]
39. Amarasinghe, K.; Kenney, K.; Manic, M. Toward Explainable Deep Neural Network Based Anomaly Detection. In Proceedings of the 2018 11th International Conference on Human System Interaction (HSI), Gdańsk, Poland, 4–6 July 2018; pp. 311–317.
40. Antwarg, L.; Miller, R.M.; Shapira, B.; Rokach, L. Explaining Anomalies Detected by Autoencoders Using SHAP. *arXiv* **2019**, arXiv:1903.02407.
41. Chen, X.; Deng, L.; Huang, F.; Zhang, C.; Zhang, Z.; Zhao, Y.; Zheng, K. DAEMON: Unsupervised Anomaly Detection and Interpretation for Multivariate Time Series. In Proceedings of the 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 19–22 April 2021; pp. 2225–2230.
42. Haldar, S.; John, P.G.; Saha, D. Reliable Counterfactual Explanations for Autoencoder based Anomalies. In Proceedings of the 3rd ACM India Joint International Conference on Data Science & Management of Data (8th ACM IKDD CODS & 26th COMAD), Bangalore, India, 2–4 January 2021; pp. 83–91.
43. Xu, H.; Wang, Y.; Jian, S.; Huang, Z.; Wang, Y.; Liu, N.; Li, F. Beyond Outlier Detection: Outlier Interpretation by Attention-Guided Triplet Deviation Network. In Proceedings of the Web Conference 2021, Ljubljana, Slovenia, 19–23 April 2021; pp. 1328–1339.
44. Deiseroth, B.; Deb, M.; Weinbach, S.; Brack, M.; Schramowski, P.; Kersting, K. AtMan: Understanding Transformer Predictions Through Memory Efficient Attention Manipulation. *Adv. Neural Inf. Process. Syst.* **2023**, *36*, 63437–63460.
45. IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. Available online: https://www.unb.ca/cic/datasets/ids-2017.html (accessed on 25 January 2022).
46. Kummerow, A.; Henneke, M.; Bachmann, P.; Krackruegge, S.; Laessig, J.; Nicolai, S. Cyber-security platform for the transparent cyber-attack detection in energy supply infrastructures. In Proceedings of the ETG Congress 2023, Kassel, Germany, 25–26 May 2023; pp. 1–7.
47. Kummerow, A.; Esrom, A.; Nicolai, S.; Bretschneider, P. Transparent autoencoding of network packets with self-attention-based transformers. In Proceedings of the 2023 IEEE 48th Conference on Local Computer Networks (LCN), Daytona Beach, FL, USA, 1–5 October 2023; pp. 1–4.
48. Kummerow, A.; Monsalve, C.; Bretschneider, P. Siamese recurrent neural networks for the robust classification of grid disturbances in transmission power systems considering unknown events. *IET Smart Grid* **2021**, *5*, 51–61. [CrossRef]
49. Sundararajan, M.; Taly, A.; Yan, Q. Axiomatic attribution for deep networks. In Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia, 17 July 2017; Volume 70, pp. 3319–3328.