future internet

MDPI

Article

Deep Learning for Intrusion Detection Systems (IDSs) in Time Series Data †

Konstantinos Psychogyios [1], Andreas Papadakis [1], Stavroula Bourou [1], Nikolaos Nikolaou [1], Apostolos Maniatis [1] and Theodore Zahariadis [1,2,*]

[1] Synelixis Solutions S.A., 10 Farmakidou Av., 34100 Chalkida, Greece; psychogios@synelixis.com (K.P.); papadakis@synelixis.com (A.P.); bourou@synelixis.com (S.B.); nikolaou@synelixis.com (N.N.); maniatis@synelixis.com (A.M.)

[2] General Department, National and Kapodistrian University of Athens, 15772 Athens, Greece

* Correspondence: zahariad@uoa.gr

† This paper is an extended version of our paper published in Presents the outcomes of the 20th International Conference on Distributed Computing and Artificial Intelligence 2023, Time-Series Modeling for Intrusion Detection Systems, July 2023.

Abstract: The advent of computer networks and the internet has drastically altered the means by which we share information and interact with each other. However, this technological advancement has also created opportunities for malevolent behavior, with individuals exploiting vulnerabilities to gain access to confidential data, obstruct activity, etc. To this end, intrusion detection systems (IDSs) are needed to filter malicious traffic and prevent common attacks. In the past, these systems relied on a fixed set of rules or comparisons with previous attacks. However, with the increased availability of computational power and data, machine learning has emerged as a promising solution for this task. While many systems now use this methodology in real-time for a reactive approach to mitigation, we explore the potential of configuring it as a proactive time series prediction. In this work, we delve into this possibility further. More specifically, we convert a classic IDS dataset to a time series format and use predictive models to forecast forthcoming malign packets. We propose a new architecture combining convolutional neural networks, long short-term memory networks, and attention. The findings indicate that our model performs strongly, exhibiting an F1 score and AUC that are within margins of 1% and 3%, respectively, when compared to conventional real-time detection. Also, our architecture achieves an ∼8% F1 score improvement compared to an LSTM (long short-term memory) model.

Keywords: intrusion detection system; deep learning; time series forecasting; cybersecurity; machine learning

Citation: Psychogyios, K.; Papadakis, A.; Bourou, S.; Nikolaou, N.; Maniatis, A.; Zahariadis, T. Deep Learning for Intrusion Detection Systems (IDSs) in Time Series Data. *Future Internet* **2024**, *16*, 73. https://doi.org/10.3390/fi16030073

Academic Editors: Javier Prieto and Ricardo Alonso

Received: 24 November 2023
Revised: 18 December 2023
Accepted: 20 February 2024
Published: 23 February 2024

1. Introduction

The universality of the internet and computer networks has revolutionized the way we interact with each other, enabling information sharing and collaboration at an unprecedented scale. However, this pervasive connectivity has also created new opportunities for malicious actors to exploit vulnerabilities and gain unauthorized access to sensitive information [1]. As a result, the importance of effective intrusion detection systems cannot be overstated, and the need for proactive notification is emerging [2]. IDS stands for a hardware device or software application used to monitor and detect suspicious network traffic and potential security breaches flagging malicious activity. This monitoring takes place at the packet level and, thus, such a system can distinguish malicious from benign packets. Traditionally, this component has been implemented as a firewall and later as a rule-based expert system. Due to the rise in ML in recent years [3–6], state-of-the-art approaches are based on ML technologies applied to data logs from IDSs, to classify packets as suspicious or not [7–11].

Moreover, multivariate time series prediction [12–14] is a sophisticated analytical approach that involves forecasting future values of multiple interrelated variables over time. Unlike univariate time series analysis, which focuses on a single variable, multivariate time series prediction considers the dynamic interactions and dependencies among several variables simultaneously. This method is particularly relevant in fields such as finance [15–17], weather forecasting [18], and industrial processes [19], where various factors influence the outcome of interest. The complexity lies in capturing the intricate relationships among different variables and understanding how changes in one can impact the others. Advanced machine learning techniques, including recurrent neural networks (RNNs) [20], LSTMs, and autoregressive integrated moving average (ARIMA) models [21], are commonly employed to handle the complexity of multivariate time series data. Accurate predictions in this context can provide valuable insights into informed decision-making, risk management, and optimizing processes in diverse domains.

Machine learning IDSs can be broadly categorized into two types: (i) classification-based [22–24] and (ii) anomaly-based [25]. Classification-based IDSs use machine learning algorithms to classify incoming data into different categories based on a set of features. Even though classification-based IDSs are effective in detecting known attacks, they can be less effective in identifying new and unknown attacks that have a small correlation with the training dataset. On the other hand, anomaly detection-based approaches use statistical models and machine learning algorithms to establish a baseline of normal behavior and identify deviations from that baseline. Unlike classification-based IDSs, anomaly-based IDSs can detect unknown or novel attacks that have not been previously seen. However apart from this advantage, these models cannot easily specify the type of attack and perform worse than classification approaches for known data types [26–28].

The IDS methodologies discussed above suffer from a significant drawback, namely the process of identifying and categorizing anomalies takes place in real-time. To address this issue, the present study suggests a proactive intrusion detection system (i.e., after the attack has taken place) that can detect and isolate malevolent packets prior to their entry into the system. This model is a novel combination of a convolutional neural network (CNN), LSTM, and attention (ATT) that can handle complex data and identify strong patterns. To achieve this, the proposed model utilizes a window of W preceding packets to predict the existence of an attack in the subsequent T packets (prediction window), allowing for the classification of the upcoming behavior in advance and enabling the application of security measures. To evaluate the effectiveness of this approach, we conducted experiments on the UNSW-NB15 dataset [29], a widely used benchmark for evaluating intrusion detection systems. Our model achieved an F1 score of 83% for the $T = 1$ case, which is similar to a real-time IDS classification. Moreover, it attained an F1 score of 91% for predicting the existence of an attack in the next $T = 20$ packets.

Our main contributions can be summarized as follows:

- We extend and improve existing intrusion detection systems by implementing proactive prediction.
- We compare our approach with state-of-the-art current methodologies.
- We experiment with the configuration parameters (e.g., size of the input window) of our method thoroughly to provide insights.
- We propose a novel methodology combining deep learning components.

The current manuscript is an extension of the previous work presented at the 2023 International Symposium on Distributed Computing and Artificial Intelligence (DCAI) [30], providing a more complex and robust architecture with more experiments, as well as an ablation study to validate the proposed architecture.

The structure of the remaining sections is as follows: Section 2 provides an overview of the previous approaches on IDSs based on machine learning as well as multivariate time series prediction state-of-the-art methodologies. Section 3 provides an overview of the IDS dataset with the relevant prediction features. In Section 4, we delve into the methodology behind the proactive IDS, the pre-processing steps we conducted, as well

as the architecture of the model. Section 5 presents the results of our experiments, which evaluate the performance of our approach in different scenarios with many alternative hyperparameter settings. In Section 6, we perform an ablation study to validate the architectural choices of the model. Lastly, in Section 7, conclusions are drawn, and potential future directions for further research are outlined.

2. Related Work

2.1. Machine Learning Intrusion Detection Systems

The field of IDSs using machine learning has seen extensive research with new methods and datasets emerging frequently [31,32]. Predicting attacks through IDS log analysis can serve as a proactive security notification feed for an organization, enhancing complementary analysis and the assessment of vulnerabilities, as pursued in [33]. Maseer, Z.K et al. [34] evaluated many machine learning classification methods on the CIC-IDS2017 [35] dataset. Regarding the pre-processing steps, they conducted one-hot encoding and normalization. They also employed parameter tuning and k-fold cross-validation for the training phase. The methods employed were the standard classification approaches, such as random forest, support vector machines, convolutional neural networks, etc., for the binary classification task. They measured accuracy (with binary accuracy, F1 score, precision, etc.) and training/testing times. The results showed that KNN, random forest, and naive Bayes achieve excellent results for these metrics.

Imran, M. et al. [36] evaluated custom autoencoder-based models against KDD-99 [37] for the multiclass classification problem. They developed a non-symmetric deep autoencoder, which was either used as a single model (NDAE) or in a stacked manner (S-NDAE). The term non-symmetric refers to the architectures of the encoder and decoder models, which in this case are not similar (symmetric). They evaluated the performance of these models with common metrics, namely accuracy, precision, etc. The results showed that this method achieves better metrics compared to different state-of-the-art approaches. Saba, T. et al. [38] developed an intrusion detection model that was tested with BoT-IoT [39] and NID datasets (https://www.kaggle.com/datasets/sampadab17/network-intrusion-detection 23 November 2023). The proposed model was a convolutional neural network. The BoT-IoT dataset was used for the multiclass classification task, whereas NID was used for binary classification. The results showed that the proposed model was able to classify the packets with an accuracy of 95%.

Pranto, M.B et al. [40] tested many classification methods using the KDD-99 dataset. Regarding the pre-processing steps, they emphasize feature selection using famous techniques (selecting K-Best) to achieve better accuracy in the classification task. From the tested methods, random forest performed the best, reaching an accuracy of 99% for the binary classification task.

Tahri, R. et al. [41] compared many articles that proposed IDSs based on the UNSW-NB15 dataset, and more specifically, on the Kaggle 100.000 sample version. In their survey, they found that random forest was the best-performing model in most of the studies, reaching an accuracy of up to 98%, specificity of up to 98%, and sensitivity of 94% for the binary classification task.

Regarding approaches that address this problem as a time series instance, Duque A. S. et al. [42] proposed the use of machine learning-based intrusion detection techniques or analyzing industrial time series data. The paper evaluated three different algorithms, namely matrix profiles, seasonal autoregressive-integrated moving average (SARIMA), and LSTM-based [43] neural networks, using an industrial dataset based on the Modbus/TCP protocol. The paper demonstrated that the matrix profile algorithm outperformed the other models in detecting anomalies in industrial time series data, requiring minimal parameterization effort.

2.2. Multivariate Time Series Prediction

When reviewing related work on multivariate time series prediction, researchers have explored various methodologies to enhance forecasting accuracy and address the complexities inherent in analyzing multiple interrelated variables over time [44–46]. Additionally, existing literature has investigated diverse applications of multivariate time series prediction, ranging from financial markets to healthcare, contributing valuable insights into the challenges and advancements within this interdisciplinary field.

Bloemheuvel, S. et al. [47] introduced a novel graph neural network (GNN)-based architecture, TISER-GCN, for multivariate time series regression, in the context of sensor networks. It addresses the limitations of existing deep learning techniques that focus solely on time series data, neglecting spatial relations among geographically distributed sensors. The proposed model was evaluated using high-frequency seismic data, demonstrating its effectiveness when compared to baseline models and traditional machine learning methods, with contributions including the development of a flexible architecture for various use cases, a thorough evaluation of diverse seismic datasets, and a systematic analysis of the model's capabilities through extensive experimentation.

Gorbett, M. et al. [48] proposed an extension of the lottery ticket hypothesis to time series Transformers, demonstrating that pruning and binarizing the weights of the model maintains accuracy similar to that of a dense Transformer. Employing the Biprop algorithm, a technique proven on complex datasets, the combination of weight binarization and pruning was applied to achieve computational advantages, reducing non-zero floating-point operations (FLOPs) and storage sizes. The approach was specifically tested on multivariate time series modeling, showcasing its effectiveness in tasks like classification, anomaly detection, and forecasting, with potential applications in resource-constrained environments such as IoT devices, engines, and spacecraft.

Wang, D. et al. [49] addressed the importance of accurate predictions in various applications of multivariate time series, such as stock prices, traffic prediction, and COVID-19 spread forecasts. They introduced the challenges faced in capturing both temporal relationships and variable dependencies in existing forecasting methods, emphasizing the need for a comprehensive understanding of underlying patterns. The work proposed a spatiotemporal self-attention-based LSTNet model, integrating spatial and temporal self-attention mechanisms to capture relationships among variables and historical observations. The contributions included the effectiveness of the proposed model in capturing spatiotemporal relationships, a novel objective function to address imbalanced errors among variables, and extensive experiments demonstrating the efficiency of LSTM-based methods in multivariate time series forecasting.

2.3. Findings and Contributions

From the aforementioned state-of-the-art review, we see that most researchers have implemented the intrusion detection system within the classification framework. Even though this approach is effective, it has limited application since the inference happens in real-time and not proactively. We also see that multivariate time series prediction is an area of active research where new deep learning approaches are frequently proposed for various tasks of different thematic areas. From the related work review, we observe that some researchers have tried to frame this problem as a time series prediction with promising results. When it comes to the UNSW-NB15 dataset, most researchers choose an incomplete version of the dataset (approx. 100.000 samples instead of $\sim$2.5 million samples). This dataset is tailored (unbalance is removed, only samples with strong correlation with the target label are kept) to machine learning applications and is not a representative version of real-world scenarios. For the reasons above, our work differs significantly since we (i) use the whole version of the dataset to emulate real-world applications, (ii) frame the intrusion detection system as a time series problem, and (iii) conduct multiple experiments to thoroughly compare against the common approaches.

3. Dataset

The dataset we used to validate our approach is UNSW-NB15, which was created by researchers at the University of South Wales in Australia. This dataset was produced by the IXIA PerfectStorm where raw packets were gathered for common attacks such as fuzzers, analysis, backdoors, and denial-of-service (DoS). We selected this dataset primarily due to its extensive use in research studies, serving as a standard benchmark for comparing various network intrusion detection systems. Additionally, it includes a time feature, enabling the conversion of the problem into a time series context.

The UNSW-NB15 dataset contains approximately 2.5 million samples of network packets and 49 corresponding labels (48 features and 1 label). These features can be viewed in Table 1. From these features, we employ 44 after the pre-processing steps. The features include basic information such as source and destination IP addresses, as well as more advanced features such as the packet size, time to live, and protocol type. This dataset also contains time-based features, namely the starting and ending times of a packet, which helps us sort and convert the samples to a time series format. Concerning the labels (binary), the dataset consists of 2,218,760 benign packets and 321,283 malicious. The mean number of packets involved in the interactions between specific entities (IPs) in this dataset is 8167. Also, the mean time between two consecutive packets is 1 second.

Table 1. Feature Table.

Attr.	Feature	Attr.	Feature	Attr.	Feature
1	srcip	17	Spkts	33	synack
2	sport	18	Dpkts	34	ackdat
3	dstip	19	swin	35	is_sm_ips_ports
4	dsport	20	dwin	36	ct_state_ttl
5	proto	21	stcpb	37	ct_flw_http_mthd
6	state	22	dtcpb	38	is_ftp_login
7	dur	23	smeansz	39	ct_ftp_cmd
8	sbytes	24	dmeansz	40	ct_srv_src
9	dbytes	25	res_bdy_len	41	ct_srv_dst
10	sttl	26	Sjit	42	ct_dst_ltm
11	dttl	27	Djit	43	ct_src_ltm
12	sloss	28	Stime	44	ct_src_dport_ltm
13	dloss	29	Ltime	45	ct_dst_sport_ltm
14	service	30	Sintpkt	46	ct_dst_src_ltm
15	Sload	31	Dintpkt	47	attack_cat
16	Dload	32	tcprtt	48	trans_depth

4. Methodology

In this section, we describe the proposed methodology for proactive notification based on IDS logs. We describe the pre-processing steps as well as the model we used for the prediction. An overview of the whole system can be viewed in Figure 1.

The architecture consists of three main components: packet input, pre-processing, and ML model.

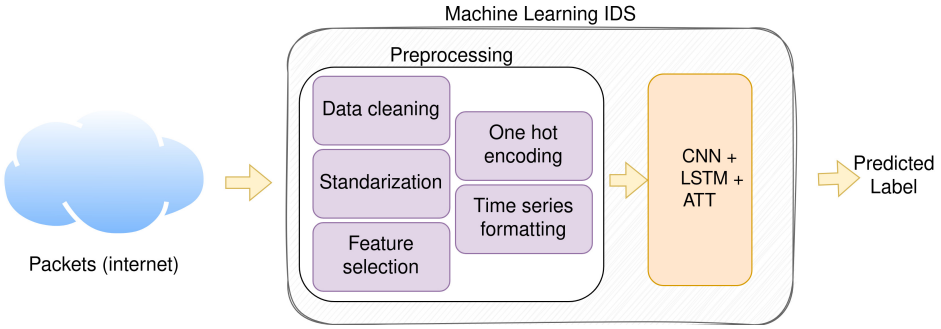

Figure 1. Methodology for the proactive IDS.

4.1. Preprocessing

Before feeding the dataset to the model, we first convert it into the right format. Thus, we initially **clean the data,** discarding *NaN* and corrupted values. The latter could be unrealistic attribute values, such as the negative packet length or unrealistic values that are too far from the feature mean (outliers). Such errors could be due to feature extraction tool errors or transportation losses.

The second pre-processing step is **data scaling**, using the min-max scaler. Scaling is the process of transforming the values of numeric variables in a dataset to a common scale. Since neural networks tend to calculate the distance between data points, we do not want features with larger scales to dominate the ones with lower scales.

The third pre-processing step is **feature selection**, which is performed using the ANOVA (analysis of variance) *F*-value. This is a statistical measure that assesses whether the means of two or more groups are significantly different from each other. The mathematical formula for this is as follows:

$$F = \frac{MS_{between}}{MS_{within}} \tag{1}$$

where for $MS_{between}$ (mean squares between):

$$MS_{between} = \frac{\sum_{i=1}^{k} n_i(\bar{X}_i - X)^2}{k-1} \tag{2}$$

and for MS_{within} (mean squares within):

$$MS_{within} = \frac{\sum_{j=1}^{n} (X_{ij} - \bar{X}_i)^2}{k_n - k} \tag{3}$$

For these equations, n_i is the number of observations in the i-th group, $\bar{X}_i$ is the mean of the i-th group, $\bar{X}$ is the overall mean, k is the number of groups, and n is the number of observations within each group. This statistic is computed between all features and the label.

The fourth pre-processing step is **one-hot encoding** since we have many categorical features that can take multiple values. This technique gives a column with N different values and creates N different binary columns, where 1 indicates the presence of this value in the current instance and 0 indicates the absence. An example is the categorical variable 'service', which indicates the type of packet and can be 'dns', 'http', 'smtp', etc.

The final stage of pre-processing involves **formatting** the data into a **time series** format. To accomplish this, we begin by sorting the dataset based on the time (starting time) feature. During this pre-processing phase, there is an option to either generate time series objects for the entire sorted dataset or initially group the dataset by (sender IP and receiver IP) pairs, subsequently creating time series objects using a specific pair of IPs. The former analyzes the overall network traffic, while the latter specifically monitors the communication flow between two designated entities. Next, using the sorted dataset, we generate windows

comprising W time points and a label, where W is the size of the input window. Here, the initial W points serve as the input to the model, and the label is the target that must be predicted. For each of the W input points, we keep all the features (not only the labels) to formulate the problem of multivariate time series forecasting. The label is calculated based on a value T and indicates the existence of an attack in the forthcoming T packets. For example, if $W = 10$ and $T = 5$, and the labels of the T packets are $[01001]$, the label of the time series instance would be 1 because there is a 1 in the T vector. Otherwise, it would be 0. Also, in such a case, we would use ten consecutive instances of the dataset as input for the model.

We also incorporate overlapping between the different windows to utilize the data to their fullest potential.

4.2. ML Model

In this subsection, we describe the architectural choices for the machine learning intrusion detection module. This model consists of a convolutional part followed by the time series processing part. A complete overview of the architectural choices is displayed in Figure 2.

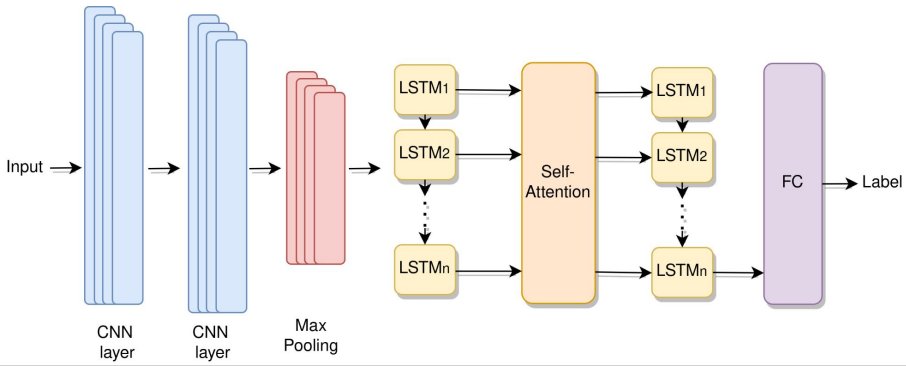

Figure 2. Architecture of our multivariate time series prediction model.

First, our model uses two convolutional layers that work on the feature axis. These layers have a dual purpose: they compress the feature space and merge information. This is crucial because the original dataset contains 44 features, some of which may be less useful or relevant. To address this, we employ a convolutional layer to identify and enhance important information during training. The output then passes through a max pooling layer to further emphasize significant feature aspects. It is worth noting that the time steps remain unchanged, and the convolution only affects the feature axis. As a result, we obtain a time series object with fewer features while preserving the same number of time steps.

Let X be the input time series with N time steps and D features. The output Y_{conv} after the convolutional layers and max pooling can be represented as follows:

$$Y_{conv} = MaxPool(Conv_2(Conv_1(X))) \quad (4)$$

Then, the modified time series is passed into the time module, which comprises two LSTM layers with attention incorporated in between. The initial LSTM layer processes the time series object and concatenates information. However, due to the persistence of dense and information-rich input, we opt to not merely utilize the concatenated output of the LSTM; instead, we collect the outputs of each time step (LSTM cell). The output of the first LSTM is a time series with the same number of time steps as the input of the LSTM, where each timestep has accumulated information from previous states. Consequently, we feed the time series produced by the first LSTM into an attention layer. This attention layer operates as a self-attention module, assessing the correlation between all LSTM cell outputs, in other words, between all the time steps within the time series. This approach aids the model in concentrating on important information from previous time steps, which proves valuable

in predicting potential attacks. The resultant attention map, denoted as A, in conjunction with the output from the first LSTM layer (all cell outputs H_1), are fused and presented as input to the second LSTM layer, as follows:

$$H_{att} = Attention(H_1) \tag{5}$$

$$H_{LSTM_2} = LSTM_2([H1, A]) \tag{6}$$

With knowledge of the attention map at each time step, this second layer can effectively interconnect information, further enhancing its ability to spotlight significant past events within the time series object.

Lastly, the output of the second LSTM (only the last accumulated state) is fed to a fully connected layer, which produces a binary label, indicating the presence of an attack:

$$Y_{final} = FC(H_{LSTM_2}) \tag{7}$$

5. Experiments

This section outlines the experiments conducted to validate the approach presented in Section 4. We describe the experimental setup, show the used metrics, and demonstrate results. Then, we conduct individual experiments to test what is the best time series grouping approach, the optimal window, W, and the prediction horizon, T.

5.1. Experimental Set-Up

Prior to conducting experiments on our model, we engage in feature selection and present outcomes for various feature sets.

To utilize our dataset, we employ 5-fold cross-validation and average the results across all 5 folds. We note that after the time series formatting—instead of individual samples—we have windows of size W and, thus, the shuffling and splitting to folds are performed on such independent windows. To demonstrate the robustness of our proposed approach, we make a comparison with the standard binary IDS classification approach. We also show the effect of the input window size on the results and, thus, exhibit the change in accuracy while the value of this variable is ascending.

Apart from experiments concerning the input window size, we also experiment with the prediction horizon. Based on the aforementioned methodology, we still predict a single label, which is not always the next packet. For each window, we create a new label that indicates the existence of an attack in the next T time steps.

To ensure the effectiveness of our predictive models, we evaluate their performance using the F1 score, precision, and recall. These metrics are necessary due to the imbalanced nature of our dataset, where simple accuracy could produce inaccurate results. F1 score, precision, and recall provide a more accurate measure of the model's performance by accounting for the imbalanced dataset and ensuring a fair evaluation.

We train our models for 40 rounds, applying early stopping with a patience parameter of $p = 5$. All the models are trained using the Adam Optimizer with a learning rate of $lr = 0.001$, a weight decay of $w = 1 \times 10^5$, and a batch size of $B = 32$. A summary of the hyperparameter setup is shown in Table 2.

Table 2. Hyperparameter configuration.

Hyperparameter	Value
Weight decay (w)	1×10^5
Batch size (B)	32
Learning rate (lr)	0.001
Patience (p)	5
Epochs	40
Optimizer	Adam

5.2. Feature Selection

In this section, we present the statistical outcomes of the applied feature selection technique and perform tests using various feature sets, measuring both predictive performance and training time in seconds. The results of the ANOVA F-test are displayed in Figure 3.

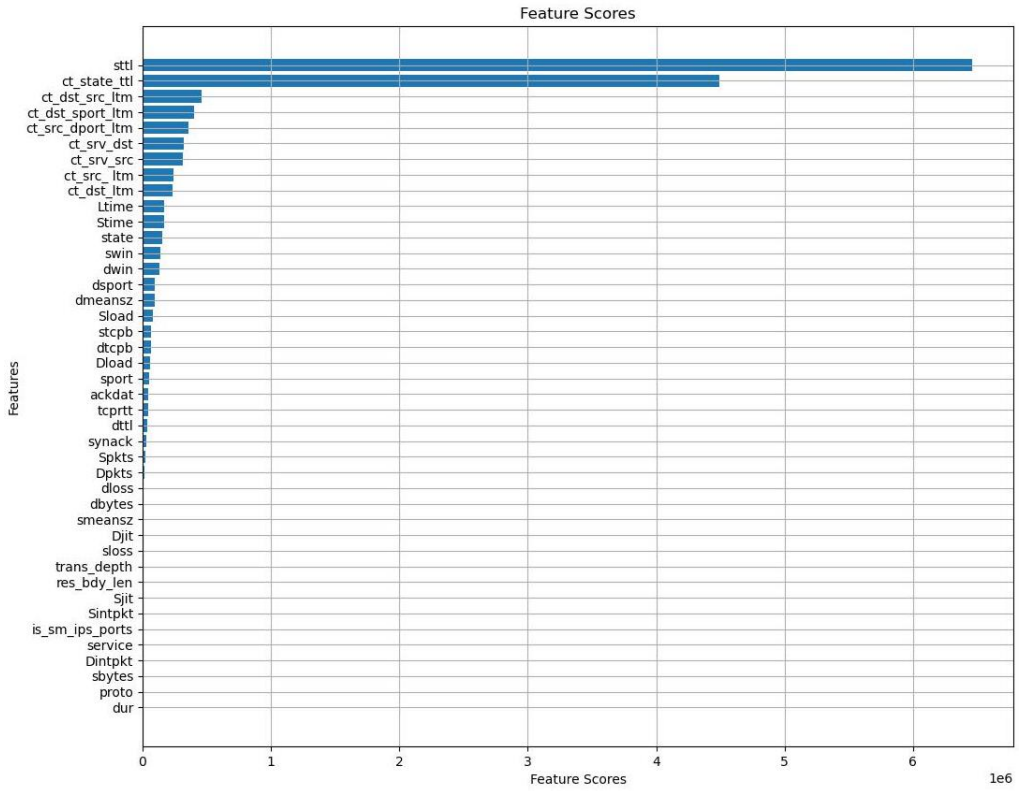

Figure 3. Feature selection results.

The features are arranged in order of relevance, and upon examining the figure, we note that 15 features exhibit significantly greater importance than the others. As a result, we generate three subsets, denoted as $S1$, $S2$, and $S3$, where $S1$ comprises all the features of the dataset, $S2$ encompasses the top 15 features determined by the statistical test, and $S3$ contains the top 10 features. Using a window $W = 50$ and a predictive horizon $T = 1$, we present the results in Table 3.

Table 3. Results for different subsets of features.

Subset	F1 score	Training Time (s)
$S1$	0.83	10
$S2$	0.81	6
$S3$	0.76	4.5

We notice a reduction in training time with a smaller subset of features, as expected, given that the overall model has fewer parameters. This also leads to a more memory-efficient model. However, the decrease in features corresponds to a decline in accuracy, which is reasonable since patterns are discarded with fewer features. Consequently, there exists a natural trade-off between predictive robustness and model complexity, and this trade-off should be customized based on a specific use case. For the following experiments, subset $S2$ is used.

5.3. Flow-Based vs. Network Grouping

Within this subsection, we examine the impact of diverse grouping approaches on the dataset to determine whether broader network information plays a crucial role in predicting forthcoming attacks or if it suffices to monitor the traffic between specific entities within the network. The grouping process is executed exclusively for the training set, leaving the testing set unchanged (i.e., the time series objects of the test set are created with network grouping). The input window W is set to 50 while the prediction horizon T is configured to 1. Specifically, we generate time series objects for the training set through the following:

1. **Network grouping (NG):** dataset sorting based on time features and subsequent creation of time series objects.
2. **Flow grouping (FG):** grouping the dataset based on IP pairs, followed by sorting each group and subsequently creating time series objects within each group.

The results of these experiments can be viewed in Figure 4.

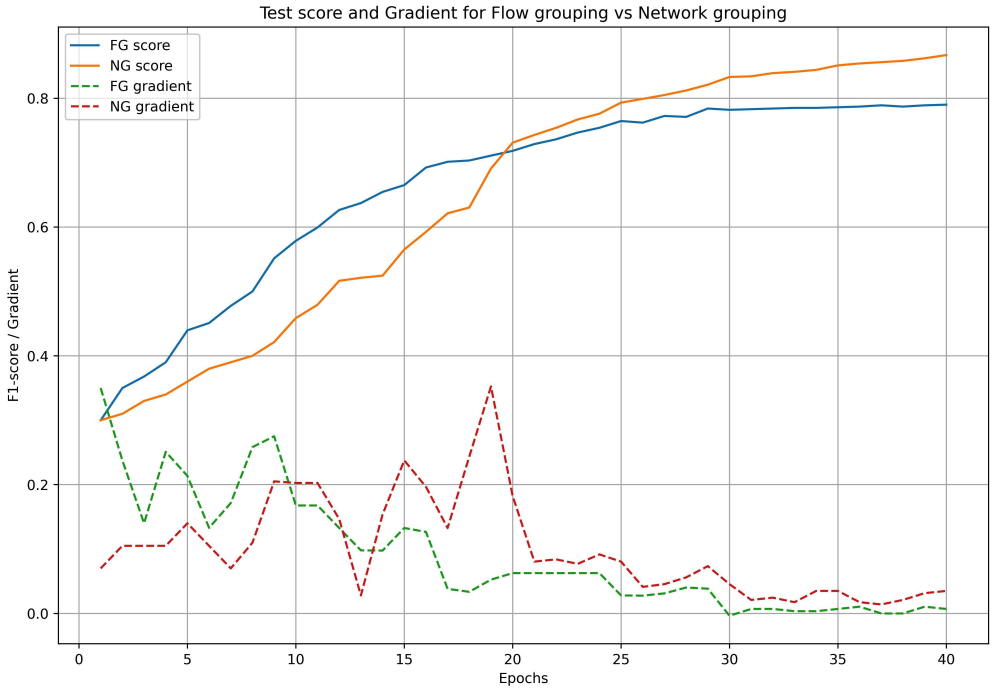

Figure 4. Comparison of different grouping approaches.

In our analysis, we observe significant differences in the results obtained from the two distinct approaches. Initially, the flow-based method exhibits a higher F1 score, with an up to 10% disparity at a certain point in, approximately, the first 10 rounds. This discrepancy arises because the pattern identification task is less intricate; correlations are more readily discernible in individual flows between two IPs.

However, the F1 score stabilizes after epoch 20 for the flow-based approach when compared to the network-based approach. After 40 epochs, the difference is $\sim$7%. Analyzing the evolution of gradients reveals that during the initial 20 epochs, both approaches exhibit steeper gradients, signifying rapid changes. Beyond epoch 20, a gradual decrease in

gradients is observed, indicating that the model is making increasingly smaller updates to its parameters as the training progresses. Notably, the gradient plot for the NG (network grouping) consistently maintains higher values compared to FG (flow grouping), suggesting a greater learning capacity for the former grouping strategy.

To further compare these two grouping approaches, we also demonstrate the receiver operating characteristic (ROC) curve for the test set predictions at epoch 40 in Figure 5.

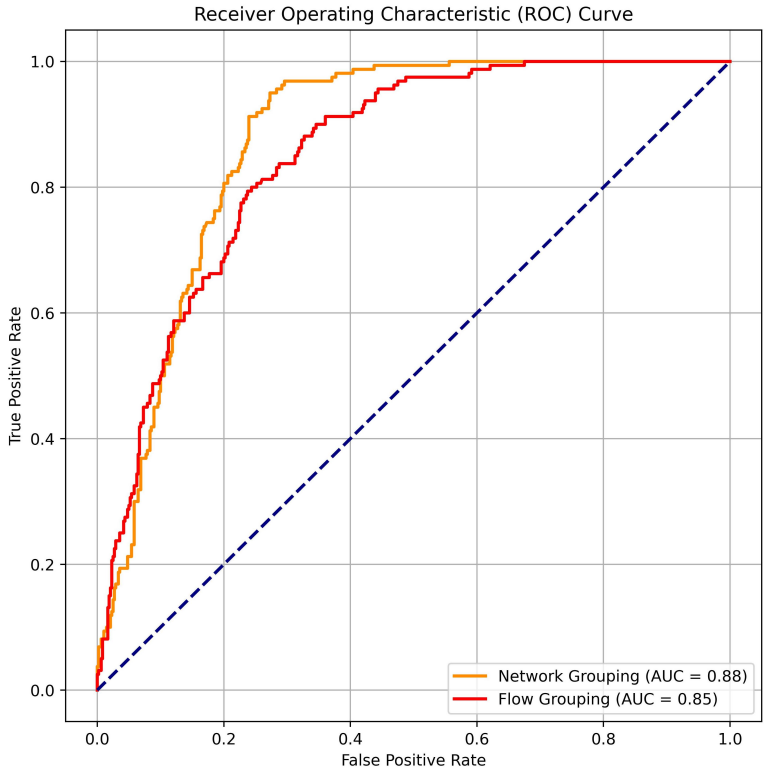

Figure 5. ROC curves for flow-based vs. network-based grouping.

The AUC score of the network grouping is better (0.88) compared to the flow grouping (0.85), suggesting a better fit of the model. We also observe that the network-based curve is closer to the upper-left corner of the plot, which generally indicates better overall performance. This implies a higher true positive rate and a lower false positive rate across different threshold settings.

The overall network-based grouping appears superior, which is attributed to the fact that the dataset was collected within the same network, and information regarding impending attacks may manifest in locations beyond the attack itself. For instance, an attacker might assess the defenses of various entities within a network before deciding to target a specific node. Therefore, this experiment suggests that valuable information about impending attacks is embedded in the overall traffic of a network, and monitoring it can contribute to a model that is more adept at predicting the presence of attacks.

5.4. Impact of W in Model Performance

Firstly, we want to test the effect of the window size on the model's accuracy. For this reason, we predict the label of the $W + 1$ packet using a window size (W) of ascending size. For this experiment, we use network-based grouping. We also want to test these cases against the classic, real-time binary classification to compare the efficiency of the two different approaches. For this experiment, the value of T is fixed, with $T = 1$. With this value fixed, after the time series formatting, we end up with 2,540,033 time series objects with 2,540,043 initial instances. This is expected since we have overlapping time steps and

we also have −10 instances for the first time series object. The label distribution is the same as the original dataset. The results can be viewed in Table 4.

Table 4. The results of the ascending input window size.

Model	F1 Score	Precision	Recall
Classification	0.85	0.90	0.78
Time Series (W = 1)	0.72	0.80	0.68
Time Series (W = 25)	0.77	0.80	0.77
Time Series (W = 50)	0.83	0.86	0.80
Time Series (W = 100)	0.83	0.86	0.81
Time Series (W = 200)	0.83	0.86	0.82

A simple (real-time) classification scenario yields marginally better results compared to the time series case, which is unsurprising, given that making classifications using present variables is substantially easier than making predictions about future events. Specifically, the F1 score of the classification method is only 1% higher than that of the time series model for $W = 50$, which is deemed acceptable. Additionally, the study indicates that an increase in the input window size enhances model accuracy. This is because utilizing more samples generally leads to improved predictions, by providing more data for analysis. Nevertheless, this also entails a greater computational burden, as a larger window entails more parameters for the model. More specifically, the study demonstrates that the metrics when W is increased from $W = 1$ to $W = 50$, with a 5–6% increase for each case. No further significant enhancement in the metrics is observed between $W = 50$ and $W = 200$, implying that all relevant information for the prediction has already been exploited considering that the lifetime of any interaction between an actor and the IDS is briefer). Consequently, the methodology described herein involves a natural compromise between computational complexity and accuracy, necessitating customization for each unique application.

We also present ROC curves for the IDS classification and the time series approach with $W = 50$ in Figure 6.

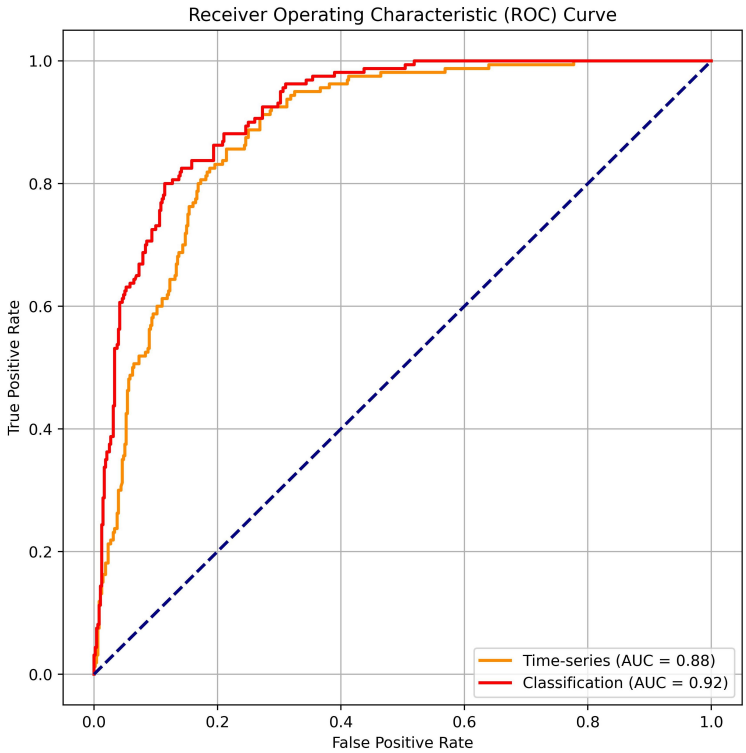

Figure 6. ROC curves for classification vs. IDS time series.

In general, we observe slightly improved outcomes for the classification scenario, as anticipated as our approach lacks information about the packet whose label is predicted. Nevertheless, upon examining both the ROC curve and the AUC score, we note that the disparity is minimal. Subsequently, our method remains viable, supporting proactive security measures.

5.5. Impact of T in Model Performance

The results related to predictions that cover a longer time frame (different values of T) are displayed in Table 5 for a fixed window size $W = 50$.

Table 5. Results for ascending prediction horizon with label distributions.

Time Steps	Labels	F1 score	Precision	Recall
$T = 1$	0:87%, 1:13%	0.83	0.86	0.80
$T = 10$	0:75%, 1:25%	0.85	0.89	0.82
$T = 20$	0:63%, 1:37%	0.91	0.92	0.91
$T = 30$	0:54%, 1:46%	0.93	0.92	0.94

The table reveals that the metrics exhibit improvement as the predictive horizon expands. This trend is expected since a larger value of T leads to more general predictions, with the focus being on detecting the presence of an attack in the upcoming T packets, rather than predicting the label of a particular packet. The most significant increase is observed between $T = 10$ and $T = 20$, where the F1 score improves by 6%, while precision and recall increase by 3% and 9%, respectively. As the value of T increases, the improvements in the metrics diminish, with only a 2% enhancement observed between the subsequent T values. We also display the label distributions for each of these cases. It is evident that with an increase in the value of T, the model achieves greater performance, as one would anticipate due to the higher probability of an attack occurring within a larger window frame. Nevertheless, anticipating an attack when using a larger value for T yields reduced information and results in sparser identification of attacks.

6. Model Architecture Ablation Study

In this section, we conduct a sequence of experiments to affirm the architecture outlined in Section 4.2. The primary objective of this investigation is to validate the model's architecture and substantiate that the chosen design elements are accurate and well-considered. To achieve this, we utilize the experimental configuration outlined in Section 5.1, wherein time series objects are generated from the entire sorted dataset, and the metrics are computed through cross-validation. The outcomes of this study are presented in Table 6.

Table 6. Results for different architectures of the time-series model.

Model	F1 score	Precision	Recall
LSTM	0.77	0.86	0.70
CNN	0.71	0.83	0.62
CNN + LSTM	0.81	0.90	0.75
CNN + LSTM + ATT	0.85	0.90	0.81

Here, we observe that the simple models consisting of a standalone LSTM or CNN perform the worst since they are too shallow and cannot handle complex patterns within the dataset. This is apparent for all three metrics displayed in the table. When it comes to the combination of LSTM and CNN, we observe an overall improvement in the metrics, which is due to the fact that the synergistic integration of both architectures leverages the strengths of each model. The differences from the simple LSTM are $\sim$0.4, 0.4, and 0.5 for the F1 score, precision, and recall. Finally, we see that the proposed model that adds an attention

layer between the two LSTM layers performs the best because the introduction of attention mechanisms allows the model to focus on the most relevant parts of the input sequence, enhancing its ability to capture intricate patterns and dependencies. This targeted attention mechanism mitigates information loss and facilitates more effective learning of long-range dependencies, contributing to superior performance across all evaluated metrics.

7. Conclusions

In this study, we re-examined the conventional approach of intrusion detection systems based on machine learning and redefined the problem as a time series prediction task. Our results demonstrate that our proposed methodology is capable of proactive operation rather than merely reacting to security breaches, achieving an F1 score within a margin of 1% compared to the real-time approach and an AUC score that is lower by only 3%. By embracing the time series prediction approach, we acknowledge the dynamic nature of network threats and the need for a proactive defense strategy. This shift in perspective allows us to anticipate and counteract emerging attack techniques, thereby staying one step ahead of malicious actors. Through our research, we aim to contribute to the continuous improvement of intrusion detection systems, ensuring the safety and security of our interconnected digital world. The results indicate that there is a natural trade-off between computation complexity and window size, W, so the choice must be tailored to a specific use case. Based on our experiments, there is an additional trade-off between feature sets and performance, where fewer features result in faster training/inference but also smaller accuracy. Furthermore, we reveal that analyzing the overall traffic of a network (network grouping) can uncover patterns not directly linked to specific nodes, such as those associated with IP, providing valuable insights into intrusion detection. We also test many architectural approaches and conclude with a complex time series prediction model that achieves robust metric scores.

In the future, we will aim to validate our approach further by employing more IDS datasets to further support our claims. We also plan to evaluate the performance of a model trained on one dataset but tested on another dataset with the same features to assess cross-dataset validation. Lastly, we plan to test our methodology in both centralized and federated learning paradigms since such a model would benefit from diverse data gathered from multiple devices.

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

Funding: This work was funded by the H2020 CyberSEAS project, contract no. 101020560, within the H2020 Framework program of the European Commission.

Data Availability Statement: The dataset used for this article is available online at https://research.unsw.edu.au/projects/unsw-nb15-dataset 23 November 2023.

Acknowledgments: We acknowledge the equal contributions of all authors.

Conflicts of Interest: All authors are employees of Synelixis Solutions S.A. Corresponding author declares that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MDPI	Multidisciplinary Digital Publishing Institute
DOAJ	Directory of Open Access Journals
ML	machine learning
ATT	attention
GNN	graph neural network
LSTM	long short-term memory
CNN	convolutional neural network
DoS	denial-of-service

References

1. Alshamrani, A.; Myneni, S.; Chowdhary, A.; Huang, D. A survey on advanced persistent threats: Techniques, solutions, challenges, and research opportunities. *IEEE Commun. Surv. Tutor.* **2019**, *21*, 1851–1877. [CrossRef]
2. Khraisat, A.; Gondal, I.; Vamplew, P.; Kamruzzaman, J. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity* **2019**, *2*, 1–22. [CrossRef]
3. Dou, B.; Zhu, Z.; Merkurjev, E.; Ke, L.; Chen, L.; Jiang, J.; Zhu, Y.; Liu, J.; Zhang, B.; Wei, G.W. Machine learning methods for small data challenges in molecular science. *Chem. Rev.* **2023**, *123*, 8736–8780. [CrossRef] [PubMed]
4. Psychogyios, K.; Ilias, L.; Ntanos, C.; Askounis, D. Missing value imputation methods for electronic health records. *IEEE Access* **2023**, *11*, 21562–21574. [CrossRef]
5. Psychogyios, K.; Ilias, L.; Askounis, D. Comparison of Missing Data Imputation Methods using the Framingham Heart study dataset. In Proceedings of the 2022 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI), Ioannina, Greece, 27–30 September 2022; IEEE: Piscataway, NJ, USA, 2022; pp. 1–5.
6. Psychogyios, K.; Leligou, H.C.; Melissari, F.; Bourou, S.; Anastasakis, Z.; Zahariadis, T. SAMStyler: Enhancing Visual Creativity with Neural Style Transfer and Segment Anything Model (SAM). *IEEE Access* **2023**, *13*, 100256–100267. [CrossRef]
7. Halbouni, A.; Gunawan, T.S.; Habaebi, M.H.; Halbouni, M.; Kartiwi, M.; Ahmad, R. Machine learning and deep learning approaches for cybersecuriy: A review. *IEEE Access* **2022**, *10*, 19572–19585. [CrossRef]
8. Zhu, J.J.; Yang, M.; Ren, Z.J. Machine learning in environmental research: Common pitfalls and best practices. *Environ. Sci. Technol.* **2023**, *57*, 17671–17689. [CrossRef]
9. He, K.; Kim, D.D.; Asghar, M.R. Adversarial machine learning for network intrusion detection systems: A comprehensive survey. *IEEE Commun. Surv. Tutor.* **2023**, *25*, 538–566. [CrossRef]
10. Hariharan, S.; Rejimol Robinson, R.R.; Prasad, R.R.; Thomas, C.; Balakrishnan, N. XAI for intrusion detection system: Comparing explanations based on global and local scope. *J. Comput. Virol. Hacking Tech.* **2023**, *19*, 217–239. [CrossRef]
11. Al-Shareeda, M.A.; Manickam, S.; Ali, M. DDoS attacks detection using machine learning and deep learning techniques: Analysis and comparison. *Bull. Electr. Eng. Inform.* **2023**, *12*, 930–939. [CrossRef]
12. Wang, X.; Liu, H.; Du, J.; Dong, X.; Yang, Z. A long-term multivariate time series forecasting network combining series decomposition and convolutional neural networks. *Appl. Soft Comput.* **2023**, *139*, 110214. [CrossRef]
13. Wang, J.; Lin, L.; Gao, S.; Zhang, Z. Deep generation network for multivariate spatio-temporal data based on separated attention. *Inf. Sci.* **2023**, *633*, 85–103. [CrossRef]
14. Wang, K.; Li, K.; Zhou, L.; Hu, Y.; Cheng, Z.; Liu, J.; Chen, C. Multiple convolutional neural networks for multivariate time series prediction. *Neurocomputing* **2019**, *360*, 107–119. [CrossRef]
15. Tsay, R.S. *Multivariate Time Series Analysis: With R and Financial Applications*; John Wiley & Sons: Hoboken, NJ, USA, 2013.
16. Yang, Y.; Lu, J. Foreformer: An enhanced transformer-based framework for multivariate time series forecasting. *Appl. Intell.* **2023**, *53*, 12521–12540. [CrossRef]
17. Hossin, M.S. Interest rate deregulation, financial development and economic growth: Evidence from Bangladesh. *Glob. Bus. Rev.* **2023**, *24*, 690–703. [CrossRef]
18. Sanhudo, L.; Rodrigues, J.; Vasconcelos Filho, E. Multivariate time series clustering and forecasting for building energy analysis: Application to weather data quality control. *J. Build. Eng.* **2021**, *35*, 101996. [CrossRef]
19. Yao, Y.; Yang, M.; Wang, J.; Xie, M. Multivariate Time-Series Prediction in Industrial Processes via a Deep Hybrid Network Under Data Uncertainty. *IEEE Trans. Ind. Inform.* **2022**, *19*, 1977–1987. [CrossRef]
20. Medsker, L.R.; Jain, L.C. Recurrent neural networks. *Des. Appl.* **2001**, *5*, 2.
21. Shumway, R.H.; Stoffer, D.S.; Shumway, R.H.; Stoffer, D.S. ARIMA models. In *Time Series Analysis and Its Applications: With R Examples*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 75–163.
22. Le, T.T.H.; Oktian, Y.E.; Kim, H. XGBoost for imbalanced multiclass classification-based industrial internet of things intrusion detection systems. *Sustainability* **2022**, *14*, 8707. [CrossRef]
23. Ahakonye, L.A.C.; Nwakanma, C.I.; Lee, J.M.; Kim, D.S. Agnostic CH-DT technique for SCADA network high-dimensional data-aware intrusion detection system. *IEEE Internet Things J.* **2023**, *10*, 10344–10356. [CrossRef]

24. Rabhi, S.; Abbes, T.; Zarai, F. IoT routing attacks detection using machine learning algorithms. *Wirel. Pers. Commun.* **2023**, *128*, 1839–1857. [CrossRef]
25. Hajisalem, V.; Babaie, S. A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection. *Comput. Netw.* **2018**, *136*, 37–50. [CrossRef]
26. Pang, G.; Shen, C.; Cao, L.; Hengel, A.V.D. Deep learning for anomaly detection: A review. *ACM Comput. Surv. CSUR* **2021**, *54*, 1–38. [CrossRef]
27. Yao, W.; Shi, H.; Zhao, H. Scalable anomaly-based intrusion detection for secure Internet of Things using generative adversarial networks in fog environment. *J. Netw. Comput. Appl.* **2023**, *214*, 103622. [CrossRef]
28. Xiao, J.; Yang, L.; Zhong, F.; Chen, H.; Li, X. Robust anomaly-based intrusion detection system for in-vehicle network by graph neural network framework. *Appl. Intell.* **2023**, *53*, 3183–3206. [CrossRef]
29. Moustafa, N.; Slay, J. UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In Proceedings of the 2015 Military Communications and Information Systems Conference (MilCIS), Canberra, Australia, 10–12 November 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 1–6.
30. Psychogyios, K.; Bourou, S.; Papadakis, A.; Nikolaou, N.; Zahariadis, T. Time-Series Modeling for Intrusion Detection Systems. In *International Symposium on Distributed Computing and Artificial Intelligence*; Springer Nature: Cham, Switzerland, 2023; pp. 1–10.
31. Thakkar, A.; Lohiya, R. A review on machine learning and deep learning perspectives of IDS for IoT: Recent updates, security issues, and challenges. *Arch. Comput. Methods Eng.* **2021**, *28*, 3211–3243. [CrossRef]
32. Saranya, T.; Sridevi, S.; Deisy, C.; Chung, T.D.; Khan, M.A. Performance analysis of machine learning algorithms in intrusion detection system: A review. *Procedia Comput. Sci.* **2020**, *171*, 1251–1260. [CrossRef]
33. Nikolaou, N.; Papadakis, A.; Psychogyios, K.; Zahariadis, T. Vulnerability Identification and Assessment for Critical Infrastructures in the Energy Sector. *Electronics* **2023**, *12*, 3185. [CrossRef]
34. Maseer, Z.K.; Yusof, R.; Mostafa, S.A.; Bahaman, N.; Musa, O.; Al-rimy, B.A.S. DeepIoT. IDS: Hybrid deep learning for enhancing IoT network intrusion detection. Computers. *Mater. Contin.* **2021**, *69*, 3945–3966.
35. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In Proceedings of the 4th International Conference on Information Systems Security and Privacy, Madeira, Portugal, 22–24 January 2018; pp. 108–116.
36. Imran, M.; Haider, N.; Shoaib, M.; Razzak, I. An intelligent and efficient network intrusion detection system using deep learning. *Comput. Electr. Eng.* **2022**, *69*, 107764.
37. Bay, S.D.; Kibler, D.; Pazzani, M.J.; Smyth, P. The UCI KDD archive of large data sets for data mining research and experimentation. *Acm Sigkdd Explor. Newsl.* **2000**, *2*, 81–85. [CrossRef]
38. Saba, T.; Rehman, A.; Sadad, T.; Koliv, H.; Bahaj, S.A. Anomaly-based intrusion detection system for IoT networks through deep learning model. *Comput. Electr. Eng.* **2022**, *99*, 107810. [CrossRef]
39. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Gener. Comput. Syst.* **2019**, *100*, 779–796. [CrossRef]
40. Pranto, M.B.; Ratul, M.H.A.; Rahman, M.M.; Diya, I.J.; Zahir, Z.B. Performance of machine learning techniques in anomaly detection with basic feature selection strategy—A network intrusion detection system. *J. Adv. Inf. Technol.* **2022**, *13*, 36–44. [CrossRef]
41. Tahri, R.; Jarrar, A.; Lasbahani, A.; Balouki, Y. A comparative study of Machine learning Algorithms on the UNSW-NB 15 Dataset. In Proceedings of the ITM Web of Conferences, Craiova, Romania, 29 June–2 July 2022; Volume 48, p. 03002.
42. Anton, S.D.; Ahrens, L.; Fraunholz, D.; Schotten, H.D. Time is of the essence: Machine learning-based intrusion detection in industrial time series data. In Proceedings of the IEEE International Conference on Data Mining Workshops (ICDMW), Singapore, 17–20 November 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 1–6.
43. Hochreiter, S.; Schmidhuber, J. Long Short-term Memory. *Neural Comput.* **1997**, *9*, 1735–1780. [CrossRef] [PubMed]
44. Luo, Y.; Cai, X.; Zhang, Y.; Xu, J. Multivariate time series imputation with generative adversarial networks. *Adv. Neural Inf. Process. Syst.* **2018**, *31*, 1596–1607.
45. Su, Y.; Zhao, Y.; Niu, C.; Liu, R.; Sun, W.; Pei, D. Robust anomaly detection for multivariate time series through stochastic recurrent neural network. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 2828–2837.
46. Jones, S.S.; Evans, R.S.; Allen, T.L.; Thomas, A.; Haug, P.J.; Welch, S.J.; Snow, G.L. A multivariate time series approach to modeling and forecasting demand in the emergency department. *J. Biomed. Inform.* **2009**, *42*, 123–139. [CrossRef] [PubMed]
47. Bloemheuvel, S.; van den Hoogen, J.; Jozinović, D.; Michelini, A.; Atzmueller, M. Graph neural networks for multivariate time series regression with application to seismic data. *Int. J. Data Sci. Anal.* **2023**, *16*, 317–332. [CrossRef]
48. Gorbett, M.; Shirazi, H.; l Ray, I. Sparse Binary Transformers for Multivariate Time Series Modeling. In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Long Beach, CA, USA, 6–10 August 2023; pp. 544–556.
49. Wang, D.; Chen, C. Spatiotemporal Self-Attention-Based LSTNet for Multivariate Time Series Prediction. *Int. J. Intell. Syst.* **2023**, *2023*, 9523230. [CrossRef]