Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.046607

ARTICLE

A Time Series Intrusion Detection Method Based on SSAE, TCN and Bi-LSTM

Zhenxiang He*, Xunxi Wang and Chunwei Li

School of Cyberspace Security, Gansu University of Political Science and Law, Lanzhou, 730000, China

*Corresponding Author: Zhenxiang He. Email: hzx6198@gsupl.edu.cn

Received: 08 October 2023 Accepted: 23 November 2023 Published: 30 January 2024

ABSTRACT

In the fast-evolving landscape of digital networks, the incidence of network intrusions has escalated alarmingly. Simultaneously, the crucial role of time series data in intrusion detection remains largely underappreciated, with most systems failing to capture the time-bound nuances of network traffic. This leads to compromised detection accuracy and overlooked temporal patterns. Addressing this gap, we introduce a novel SSAE-TCN-BiLSTM (STL) model that integrates time series analysis, significantly enhancing detection capabilities. Our approach reduces feature dimensionality with a Stacked Sparse Autoencoder (SSAE) and extracts temporally relevant features through a Temporal Convolutional Network (TCN) and Bidirectional Long Short-term Memory Network (Bi-LSTM). By meticulously adjusting time steps, we underscore the significance of temporal data in bolstering detection accuracy. On the UNSW-NB15 dataset, our model achieved an F1-score of 99.49%, Accuracy of 99.43%, Precision of 99.38%, Recall of 99.60%, and an inference time of 4.24 s. For the CICDS2017 dataset, we recorded an F1-score of 99.53%, Accuracy of 99.62%, Precision of 99.27%, Recall of 99.79%, and an inference time of 5.72 s. These findings not only confirm the STL model's superior performance but also its operational efficiency, underpinning its significance in real-world cybersecurity scenarios where rapid response is paramount. Our contribution represents a significant advance in cybersecurity, proposing a model that excels in accuracy and adaptability to the dynamic nature of network traffic, setting a new benchmark for intrusion detection systems.

KEYWORDS

Network intrusion detection; bidirectional long short-term memory network; time series; stacked sparse autoencoder; temporal convolutional network; time steps

1 Introduction

The burgeoning use of the internet has corresponded with a marked rise in cyberattack incidents, presenting profound risks to societies, nations, and individuals alike [1,2]. Recent data from the China Internet Emergency Center's 12th report of 2023 reveals a concerning trend: malicious software, backdoor website creations, and new security vulnerabilities are emerging at an alarming rate. Specifically, between March 20th and March 26th, there was a 52.5% jump in the spread of malicious programs from the previous interval. Backdoor website occurrences escalated by 76.2%, while the discovery of new security vulnerabilities rose by 24.9%. Every week, thousands of cybersecurity incidents demand intervention. These figures do not just spotlight the escalating cyber threat environment; they also

emphasize the growing urgency for robust network intrusion detection solutions. Numerous challenges have arisen within the emerging fields of cloud computing and the Internet of Things (IoT). Cloud computing has evolved into a modern platform marked by high scalability and on-demand services, where one of the most formidable challenges is maintaining the reliability of the network environment [3,4]. Concurrently, the exponential increase in IoT devices has precipitated a similar surge in network traffic. Even brief periods of network unavailability can have catastrophic consequences, underscoring the critical need for robust network intrusion detection systems [5,6]. Network intrusion detection empowers us to promptly identify and counteract potential security threats [7,8]. Yet, the vastness and high dimensionality of network traffic data, combined with the continuous evolution of attack techniques, question the efficacy and efficiency of conventional detection approaches [9,10]. Herein lies the crux of our research: the STL model is engineered to directly address these challenges by integrating time series analysis, thus enhancing the predictive power and responsiveness of IDS.

The digital landscape's evolution has been paralleled by an escalation in the sophistication and frequency of cyberattacks, necessitating advanced defenses in network security. Among the myriad of protective strategies, Intrusion Detection Systems (IDS) stand as sentinels against unauthorized access and malicious activities [11]. These systems not only monitor network traffic but also serve as crucial components in an organization's security architecture, ensuring the integrity and confidentiality of data.

Despite the critical role of IDS, the incorporation of time series analysis in detecting intrusions is often overlooked. Time series data—observations recorded sequentially over time—unlocks a chronological perspective, allowing security systems to discern patterns and anomalies that sporadic data points may obscure. However, the complexities of time series data and the analytical challenges it presents have led to its underutilization in intrusion detection [12,13].

Acknowledging this lacuna, our research presents the STL hybrid model—a pioneering solution that integrates time series analysis into the fabric of intrusion detection. By systematically characterizing network data over time, the STL model aims to elevate detection efficacy, especially against threats that manifest incrementally. Time series data, with its potential to unveil subtler, evolving attack patterns, serves as a linchpin in this endeavor, enhancing the predictive capabilities of IDS and reducing the incidence of false negatives.

To elucidate the novel methodology underpinning the STL model, we offer a precis of the hybrid model's components and their convergence to surmount the challenges of modern cyber threats:

- The Stacked Sparse Autoencoder (SSAE) forms the foundational layer, tasked with distilling high-dimensional data to preserve salient features crucial for subsequent analysis.
- The Temporal Convolutional Network (TCN) and Bidirectional Long Short-Term Memory Network (Bi-LSTM) synergize as the model's core, adeptly sifting through the compressed data to extract features with high temporal fidelity.

As we unfold the narrative of this paper, the subsequent sections are laid out as follows: Section 2 reviews the extant literature, shedding light on the current state of intrusion detection and the time series gap therein. Section 3 delineates the deep learning models that constitute the STL model, detailing their roles and interplay. Section 4 provides an in-depth account of the STL hybrid model's architecture and the experimental methodology we have adopted. Section 5 presents the datasets, preprocessing steps, and rigorous experimental results that validate our model's proficiency in time series analysis. Finally, Section 6 offers a synthesis of our findings, openly discusses the limitations of

our approach, and suggests directions for future research, emphasizing the broader implications of our work for the field of network intrusion detection.

Through this paper, we aim to not only bridge the current research gap but to also provide a clear roadmap for the integration of time series analysis in intrusion detection, demonstrating its criticality for robust and predictive cybersecurity defenses.

2 Related Work

2.1 Deep Learning Method

Deep learning is a subset of machine learning. Unlike traditional machine learning techniques, deep learning primarily aims to learn the inherent patterns within sample data. This approach not only streamlines model construction and feature extraction but also exhibits heightened accuracy, especially when handling large datasets. In recent times, as deep learning techniques have matured and gained traction, a growing number of researchers are harnessing these methods for network intrusion detection. Numerous systems have been crafted, integrating deep learning models like Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), Auto Encoders (AE), and Deep Belief Networks (DBN). These systems have demonstrated remarkable efficacy in identifying large volumes of anomalous data and emergent network threats [14].

CNN is inherently adept at recognizing local features of data, leading to an optimized network structure that ensures both positional and shape invariance [15]. Their capacity to perform feature extraction and reduce data dimensionality has made them increasingly prevalent in the intrusion detection field. This is largely due to their proficiency in capturing salient data feature representations. Wang et al. [16] proposed a one-dimensional CNN-based end-to-end detection classification technique. This method can autonomously learn from raw input and discern nonlinear correlations between output features. However, its reliance on one-dimensional data hampers effective local feature extraction, yielding a detection rate that falls short of the benchmark. On another front, a Long Short-Term Memory Network (LSTM) is a distinct subtype of recurrent neural networks. It possesses the capability to retain input data and foresee outputs across varying periods, effectively circumventing the gradient vanishing and exploding challenges inherent in RNNs. Consequently, LSTMs have found extensive applications, notably in the realm of natural language processing [17]. Bi-LSTM [18], an enhancement over the conventional LSTM, remedies the limitations of the latter by bridging contextual information more effectively. In 2018, Bai et al. [19] introduced the TCN. This innovative model integrates causal and dilated convolution and embeds residual connections between network layers. Such an architecture facilitates the extraction of sequential features and simultaneously staves off gradient-related issues. The ongoing refinement of deep learning algorithmic frameworks and their ensuing success across diverse sectors underscore the burgeoning potential and bright future of deep learning.

2.2 Intrusion Detection Method

Research on network intrusion detection can be traced back to the 1980s when Professor Anderson [20] pioneered the notion of "intrusion detection." Four decades hence, the principles underpinning intrusion detection have evolved, found extensive applications, and are now integral components of network security systems. A significant milestone was achieved in the 1990s when Debar and associates [21] integrated neural networks into intrusion detection, heralding the era of leveraging neural network models in this domain. Subsequently, Lin et al. [22] incorporated convolutional neural networks into network intrusion detection. Utilizing a refined LeNet-5 model coupled with

gradient descent optimization, backpropagation, and meticulous fine-tuning of learning rates, they showcased commendable classification performance on the KDD99 dataset. In a further development, Vinayakumar et al. [23] introduced an intrusion detection methodology based on an enhanced RNN. This approach tackled the gradient explosion challenge and juxtaposed its efficacy against established deep learning techniques in the realm of network intrusion detection, encompassing models like RNN and LSTM. Experimental evaluations revealed that this advanced RNN held its ground, delivering detection results on par with LSTM across datasets such as KDDCUP99, NSL-KDD, and UNSW-NB15.

Fernandez et al. [24] empirically compared machine learning and deep learning-based intrusion detection methodologies. Their findings underscored the superior classification efficacy of deep learning over traditional machine learning in the context of network intrusion detection. In another study, Radford et al. [25] utilized a Bi-LSTM-driven intrusion detection technique to classify datasets, reporting impressive detection outcomes. Thirimanne et al. [26] fashioned a real-time intrusion detection system that analyzed both inbound and outbound network data. By employing a DNN model for classifying the KDDCUP99 dataset, they achieved a striking 96% accuracy. Pooja et al. [27] harnessed a Bi-LSTM model to bifurcate the KDDCUP99 dataset, registering remarkable accuracy. Thilagam et al. [28], by fusing CNN and LSTM, embarked on multi-classifying datasets across varied attack spectrums, including botnets, resulting in enhanced classification precision compared to other established techniques. Wang et al. [29] integrated a self-attention mechanism into a Bi-LSTM framework, enhancing feature significance computation and thereby optimizing classification performance. Sinha et al. [30] proposed a detection model melding both CNN and Bi-LSTM, enabling the model to learn the spatial and temporal characteristics of data. Trials on the NSL-KDD and UNSW-NB15 datasets affirmed its superior performance over traditional machine learning and other contemporary deep learning models. Lastly, Xu et al. [31] crafted an intrusion detection model based on the Stacked Sparse Autoencoder (SSAE). Their approach involved using SSAE for data dimensionality reduction before channeling the condensed data into a softmax classifier. They further augmented the model based on SSAE, not only bolstering its intrusion detection prowess but also diminishing the rate of false alarms.

Xu et al. [32] incorporated the Logarithmic Autoencoder (LogAE) to emphasize essential data attributes, leveraging the Extreme Gradient Boosting (XGBoost) for dataset categorization. Their performance on the UNSW-NB15 dataset outshined several contemporary methods. Imran et al. [33] designed an intrusion detection system, amalgamating predictive and learning facets. With the learning segment anchored in Automated Machine Learning (AutoML) and the predictive portion hinging on the Kalman filter, evaluations on the UNSW-NB15 and CICIDS2017 datasets confirmed the augmented accuracy of this amalgamated model. Wang et al. [34] presented RUIDS, a sturdy unsupervised intrusion detection paradigm, by embedding a masked context reconstruction module within a transformer-centric self-supervised learning architecture. Trials on the UNSW-NB15 dataset validated its enhanced robustness without compromising the algorithm's efficiency. Lopez-Martin et al. [35] modernized the conventional Radial Basis Function (RBF) neural network, incorporating it into an offline reinforcement learning algorithm, delivering superior outcomes on the CICIDS2017 dataset in comparison to other models. Kim et al. [36] harnessed an LSTM-DNN framework on the primary dataset to generate mislabeled data. Employing a Generative Adversarial Network (GAN) on the retrained dataset facilitated real-time intrusion detection on the CICIDS2017 dataset without resorting to session discontinuations or data collection delays, maintaining competitive performance levels. Siddiqi et al. [37] introduced an innovative framework that synergizes an advanced feature selection mechanism with image augmentation techniques, targeting adept anomaly detection using

deep learning classifiers. The superiority of this methodology on the CICIDS2017 dataset was evident when juxtaposed against alternatives. Lastly, Deng et al. [38] proposed an avant-garde technique rooted in the Flow Topology Graph Convolutional Network (FT-GCN) tailored for intrusion detection in resource-constrained IoT networks. Benchmarking on the UNSW-NB15 dataset showcased the FT-GCN's commendable efficiency against leading-edge models.

Table 1 presents a detailed comparative analysis of the existing research on intrusion detection that we have reviewed, highlighting the unique advantages and identifying the specific challenges encountered in each referenced study [39].

Table 1: Summary of related works

Ref.	Method	Advantages	Challenges
[25]	Bi-LSTM	Identifies new malicious behaviors without prior knowledge.	Impractical full dataset training for real-world use.
[26]	DNN	Integrates a real-time feature extractor for live monitoring.	Bias towards normal data, leading to false alarms.
[27]	Bi-LSTM	Outperforms other methods in literature.	Untested for online, real-time attack detection.
[28]	CNN-LSTM	Effective combination of LSTM and CNN within an RC-NN framework.	scalability to larger network environments.
[29]	BiLSTM-Attention	Develops three new self-attention functions tailored for this detection task.	May require redefinition of distance metrics for different code smell types.
[30]	CNN-BiLSTM	The model integrates the spatial and temporal learning capabilities of CNNs and Bi-directional LSTMs.	Optimization for U2R and Worms attack categories is needed.
[31]	ISSDA	ISSDA enhances decoding precision and overall detection performance.	Requires enhanced high-dimensional data handling.
[32]	LogAE-XGBoost	Combines log learning and boosting for feature classification.	LogAE might misjudge features without standard normalization.
[33]	AutoML-KF	Utilizes ensemble methods to enhance performance.	Needs to address scalability and bandwidth handling for growing internet applications.
[34]	RUIDS	Resistant to anomaly contamination with novel reconstruction module.	Needs to address the complexity of transformer models.
[35]	RBFNN	Suitable for unbalanced and noisy datasets typical in network intrusion detection.	Needs to develop new loss functions for the RBFNN training.

(Continued)

Table 1 (continued)

Ref.	Method	Advantages	Challenges
[36]	LSTM-DNN	Enables real-time intrusion detection without waiting for session completion.	Must manage the trade-off between detection speed and accuracy.
[37]	GF-CNN	Reduces feature set for low computational overhead while maintaining precision.	Needs to identify optimal parameters for image transformation techniques like the Gabor filter.
[38]	FT-GCN	Efficiently handles limited labeling through topological structure conversion of traffic flows.	Requires optimization to improve traffic graph construction for better correlation representation.

3 Methodology

3.1 Sparse Autoencoder

AE, a derivative of the feedforward neural network, comprises three layers: input, hidden, and output [40]. Unlike feedforward networks, AE learns features in an unsupervised manner, training iteratively and adjusting connection weights through gradient descent and the backpropagation algorithm. The training objective is to preserve as much information as possible after the encoding and decoding process while ensuring that the features represented in the hidden layer possess desirable properties.

Aside from learning low-dimensional encoding, autoencoders can also learn high-dimensional Sparse Autoencoder (SAE). Assuming the dimension M of the middle-hidden layer z is greater than the dimension D of the input sample x, and making z as sparse as possible, this is known as a sparse autoencoder. The advantage of the sparse autoencoder is its high interpretability, coupled with implicit feature selection. During intrusion detection, a sparse encoder can be used to compress high-dimensional traffic data, replacing the original data with newly acquired feature samples. While preserving the original data's information, this effective compression reduces feature dimensions, enhancing model learning performance, cutting down computation, and boosting the intrusion detection speed.

By imposing sparsity constraints on the hidden layer unit z in the autoencoder, the autoencoder can learn some useful structures within the data. Given N training samples $\{x^{(n)}\}_{n=1}^{N}$, the objective function of the sparse autoencoder is

$$L = \sum_{n=1}^{N} \left\| x^{(n)} - x^{'(n)} \right\|^2 + \eta\rho\left(Z\right) + \lambda \left\| W \right\|^2 \tag{1}$$

where $Z = [z^{(1)}, \cdots, z^{(n)}]$ represents the encoding of all training samples, $\rho(Z)$ is the sparsity measurement function, and W represents the parameters in the autoencoder.

Given N training samples, the average activation value of the jth neuron in the hidden layer is

$$\hat{\rho}_j = \frac{1}{N}\sum_{n=1}^{N} z_j^{(n)} \tag{2}$$

$\hat{\rho}_j$ can be approximately viewed as the probability of the activation of the jth neuron. If we want $\hat{\rho}_j$ to be close to a pre-defined value ρ^*, for instance 0.05, the divergence between $\hat{\rho}_j$ and ρ^* can be measured using the KL distance, that is

$$KL(\rho^*||\hat{\rho}_j) = \rho^* \log \frac{\rho^*}{\hat{\rho}_j} + (1-\rho^*) \log \frac{1-\rho^*}{1-\hat{\rho}_j} \tag{3}$$

If $\hat{\rho}_j = \rho^*$, then $KL(\rho^*||\hat{\rho}_j) = 0$. Therefore, the loss function of the sparse autoencoder is

$$J_{SAE}(W) = \sum L(x, y) + \beta \sum_{j=1}^{h} KL(\rho^*||\hat{\rho}_j) \tag{4}$$

The stacked sparse autoencoder is closely related to the autoencoder, combining the stacked encoder and the sparse encoder. The features extracted from the previous layer are input to the next layer, deeply learning data features at different scales to obtain a high-level feature representation of the input data.

3.2 Temporal Convolutional Network

The Temporal Convolutional Network offers a distinctive convolutional neural network design that adapts one-dimensional convolution, making it adept at addressing time series challenges [41]. Fig. 1 depicts the comprehensive TCN architecture, encompassing causal convolution, dilated convolution, and residual connections, adeptly addressing multivariate time series extraction challenges. Its application to time series forecasting has grown in recent times. Subsequent sections detail each component's specifics.

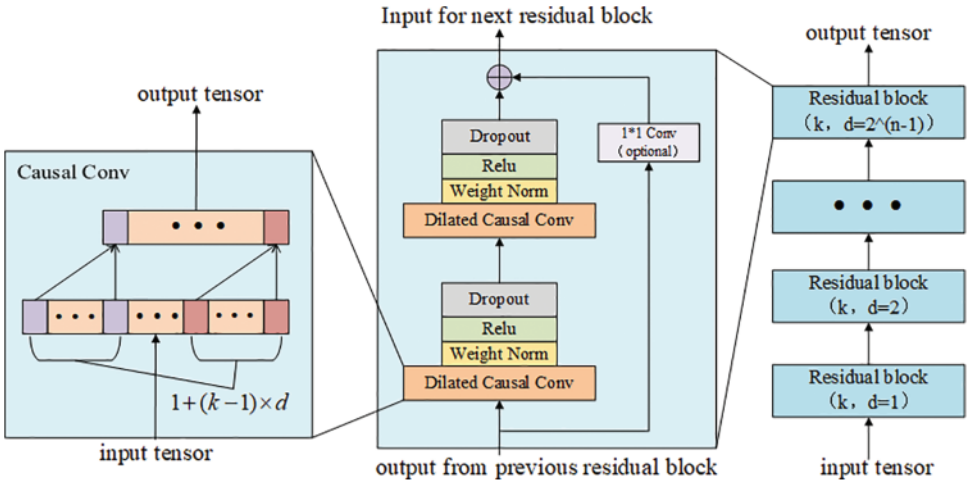


Figure 1: The complete structure of the temporal convolutional network

• Causal Convolution: Causal convolution has strict temporal constraints, using only the information from prior moments to predict the value at a specific moment. This convolution operation ensures that the TCN does not miss historical information like the CNN, thus influencing the final classification. The architecture of causal convolution is shown in Fig. 1. From Fig. 1, it can be seen that if the input time series is $= [x_1, x_2, \cdots, x_T]$, $X \in \Re^n$ and the filter is $F = [f_1, f_2, \cdots, f_T]$, the output of x_t after causal convolution is the following equation:

$$Y(T) = (X * f)(T) = \sum_{i=0}^{k-1} F(i) \cdot x_{T-i} \tag{5}$$

• Dilated Convolution: Essentially, as the network depth increases, each layer's receptive field, or the number of steps it can read, expands. This operation ensures that the size of each hidden layer is consistent with the input time step length. From Fig. 1, the output of x_T after dilated convolution with a dilated factor is the following equation:

$$Y(T) = (X *_d f)(T) = \sum_{i=0}^{k-1} F(i) \cdot x_{T-d \cdot i} \tag{6}$$

where d is the dilated factor, k is the filter size, and $T - d \cdot i$ accounts for the direction of the past.

• Residual Connections: The role of residual connections is to address the gradient vanishing problem that may occur when the network layers are too deep [42].

3.3 Long Short-Term Memory Network

LSTM is a specialized form of RNN, characterized as a Gated Recurrent Neural Network (Gated RNN). Traditional RNNs grapple with persistent dependency challenges, which LSTMs adeptly address with their gating mechanism. This makes LSTMs aptly equipped to handle extended time series datasets.

The core strength of an LSTM resides in its memory cell, which seamlessly integrates across the entire network. This design ensures minimal data loss, promoting consistent information relay during prolonged sequences. Furthermore, LSTMs leverage input, forget, and output gates to maintain gradient flow and control: the forget gate evaluates data to discard from the cell state, the input gate integrates new information into the cell state, while the output gate determines the conclusive output. A visual representation of the LSTM's recurrent architecture can be observed in Fig. 2.

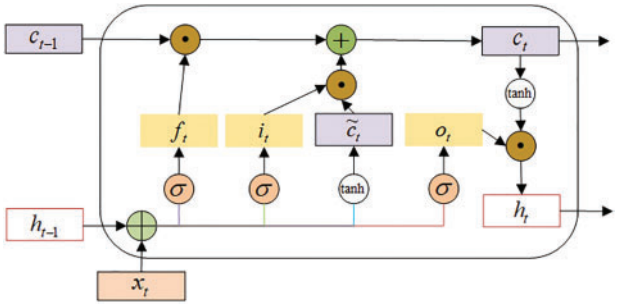


Figure 2: Recurrent unit structure of LSTM network

A Bi-LSTM is structured with dual recurrent neural network layers that share the same input but propagate information in opposite directions. Its predictions incorporate both past and future inputs. Given that the foremost layer processes data in a forward temporal sequence while the latter operates in reverse, the hidden states at a given time t are denoted as $h_t^{(1)}$ for the forward sequence and $h_t^{(2)}$ for the reverse. This relationship can be mathematically expressed as follows:

$$h_t^{(1)} = f(U^{(1)} h_{t-1}^{(1)} + W^{(1)} x_t + b^{(1)}) \tag{7}$$

$$h_t^{(2)} = f(U^{(2)} h_{t+1}^{(2)} + W^{(2)} x_t + b^{(2)}) \tag{8}$$

$$h_t = h_t^{(1)} \bigoplus h_t^{(2)} \tag{9}$$

Fig. 3 displays the bidirectional recurrent neural network unfolded over time. The working principle of LSTM typically runs from front to back according to the time sequence, learning the internal features of the data, and obtaining the final result through the classifier. In this paper, we use Bi-LSTM, which runs both forward and backward in time sequence. The final output is determined by the internal features of the classifier from two unidirectional LSTMs. The advantage of this approach is that it allows the algorithm to better learn the internal features of the data, improving classification accuracy.

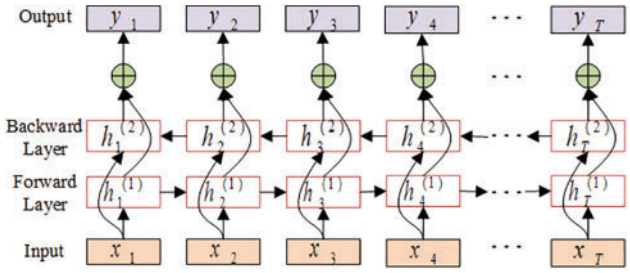


Figure 3: Bi-LSTM neural network structure expanded by time

4 Proposed Model

4.1 STL Model

We introduce a hybrid model, STL, tailored for the characteristics of time series data features, as depicted in Fig. 4. This model is structured into five distinct layers: the input layer, the SSAE layer, the TCN network layer, the Bi-LSTM network layer, and the output layer.

The input layer, being the first, inputs pre-processed data. Following this, the SSAE layer utilizes greedy layer-wise pre-training to pinpoint the optimal number of stacked layers suitable for the dataset. This model, once determined, is retained to handle the pre-processed data. The SSAE's primary function is feature reduction, ensuring the preservation of crucial data features. The SSAE achieves a concise representation of the original data features and, with the integration of sparsity constraints, diminishes noise, computational strain, and potential overfitting. Moreover, it not only maintains the integrity of reduced dimensions but also showcases its prowess in feature extraction. The processed data from the SSAE layer then becomes the input for the subsequent temporal model. The steps for SSAE feature dimensionality reduction are as follows Algorithm 1.

Algorithm 1: SSAE for Feature Dimensionality Reduction

Input: Training data $X \in \Re^{n \times d}$

Output: Trained SSAE model, Encoded data, Classifier model

Initialization: n: number of samples, d: number of features, ρ: sparsity parameter, α: weight of regularization

Step 1: Define SSAE structure:

for each layer $l \in \{1, 2, 3\}$

Encoder: Dropout layer → Dense layer with ReLU activation and sparsity regularizer.

Decoder: Dense layer with ReLU activation and sparsity regularizer.

Step 2: Define Sparse regularizer:

Regularization term for each neuron's activation a:

$$R(a) = \alpha\left(\rho \log \frac{\rho}{a} + (1-\rho) \log \frac{1-\rho}{1-a}\right)$$

Step 3: Pre-training phase:

for each layer $l \in \{1, 2, 3\}$

If $l = 1$: Input $\leftarrow X$

Else Input ←Output of encoder $l-1$

Train autoencoder l to minimize the mean squared error (MSE) between the input and the reconstructed output, incorporating the regularization term $R(a)$.

Save weights if validation loss improves.

Step 4: Initialize SSAE encoder with pre-trained weights.

Step 5: Define classification model:

Attach additional dense and LSTM layers to the SSAE encoder.

Output layer with sigmoid activation function for binary classification.

Step 6: Compile and train the classifier:

Attach additional dense and LSTM layers to the SSAE encoder.

Output layer with sigmoid activation function for binary classification.

Step 7: Evaluate the classifier:

Predict class labels for training and testing data.

Compute accuracy and false alarm rate (FAR) for both sets.

Step 8: Save the trained classifier model and encoded data.

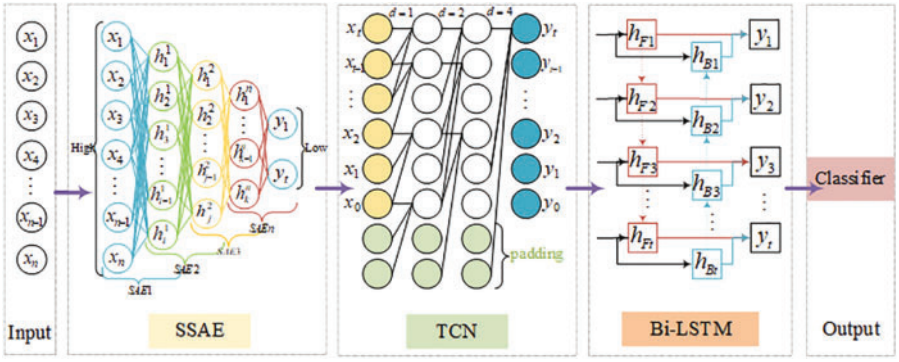


Figure 4: STL model framework

Transitioning to the third layer, we have the TCN network layer. TCN is a neural network architecture specifically designed for sequence data processing. It is characterized by its rapid convergence and superior feature extraction capabilities. This is achieved by multiple stacks of causal convolutional layers, extending the convolution's receptive field, which underscores TCN's proficiency in addressing time series issues. The fourth layer: Bi-LSTM network layer. The output of the TCN network layer serves as the input for the Bi-LSTM network layer, capturing the sequential relationships of the input. The Bi-LSTM network layer captures preceding and succeeding information of the input sequence through forward and backward loop processing. It fully learns the intrinsic patterns within the data and obtains future predictions, effectively revealing latent relationships between data. By combining TCN and Bi-LSTM, this hybrid model can fully utilize TCN's long-term memory capabilities and Bi-LSTM's bidirectional sequence-capturing abilities. This combination offers enhanced performance on complex sequences. Lastly, the fifth layer is the output layer. This layer bridges the Bi-LSTM's output to the classifier via two fully connected layers, which then executes the binary classification imperative for intrusion detection. The steps for TCN-BiLSTM for Sequential Data Processing are as follows Algorithm 2.

Algorithm 2: TCN-BiLSTM for Sequential Data Processing

Input: Sequence X of shape (time_steps, features)
Output: Trained model for sequence classification
Initialization: Set time_steps, features
Step 1: Define Model Architecture:
Input layer with shape (time_steps, features)
TCN layer with causal and dilated convolutions
Bidirectional LSTM layers
Step 2: TCN Layer
for each layer l with dilation d, filter W, input x, **do**
for each time step t **do**

$$y[t] \leftarrow ReLU\left(\sum_{k=0}^{K-1} W[K] \cdot x[t - d \cdot k]\right)$$

end for
Step 3: Bidirectional LSTM
for each time step t **do**
Compute $h_t^{forward}$, $h_t^{backward}$

$$\begin{aligned}
h_t &\leftarrow concatenate\left(h_t^{forward}, h_t^{backward}\right) \\
i_t &= \sigma\left(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}\right) \\
f_t &= \sigma\left(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}\right) \\
g_t &= \tan h\left(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}\right) \\
o_t &= \sigma\left(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}\right) \\
c_t &= f_t \cdot c_{t-1} + i_t \cdot g_t \\
h_t &= o_t \cdot \tan h\left(c_t\right)
\end{aligned}$$

where σ is the sigmoid activation function, $\tan h$ is the hyperbolic tangent function, W and b are the weights and biases of the respective gates, c_t and h_t are the cell state and hidden state at time t.

(Continued)

Algorithm 2 (continued)

Step 4: Model Compilation and Training
Compile model with optimizer, loss, metrics
Fit model on training data with TCN and BiLSTM layers
Step 5: Model Evaluation
Predict on test data using trained model
Evaluate with accuracy, precision, recall, F1-score

Fig. 4 illustrates the network structure and the arrangement of the three models within it, which is primarily determined by two critical factors.

Sequential Transfer Learning employs a structured approach: initially, SSAE reduces dimensionality and filters noise from the data, establishing a foundation for feature extraction. This is followed by the TCN layer, which is adept at capturing local temporal features within the sequence. Finally, the Bi-LSTM layers integrate the previously processed data, considering both past and future contexts, to construct a comprehensive global representation of the sequence.

Layer Complementarity is central to our architecture, with each layer designed to augment the capabilities of the others. The SSAE is the foundation, creating a compact representation of the original data. Building on this, the TCN expands the analysis into the temporal domain, utilizing the reduced feature space to examine temporal scope more effectively. Finally, the Bi-LSTM layer weaves these refined features into a cohesive timeline, capturing the complex dynamics by considering both past and future information in the sequence.

4.2 Model Experiment Process

The workflow of our hybrid model is depicted in Fig. 5. Initially, we load the original data for both training and testing from the dataset. Upon preprocessing, this data is bifurcated into distinct training and testing sets. Subsequently, 30% of these sets are randomly chosen to determine the optimal SSAE model. With each added SAE layer during the training phase, the best-performing iteration is autonomously stored. After adding a layer of SAE for training, the best-performing model is automatically saved, and the final optimal model is used in the STL hybrid model. Following this, the training set is introduced into the pre-constructed STL model for further training. After obtaining the predicted values, compare them with the original data labels separated during data preprocessing. Finally, this trained model is applied to the testing set, and its efficacy is gauged using specified evaluation metrics.

In the experiments, a time series generator is used to test the detection performance at different time steps, with its function and significance as follows:

- The Time Series Generator is essential for preparing sequential data for deep learning models. It transforms the data into a compatible format by segmenting input sequences into fixed-length samples determined by the time_steps parameter, thereby allowing the model to identify and learn from patterns within these temporal segments.
- Additionally, the generator plays a pivotal role in batch preparation during training, organizing the data into manageable units of batch_size samples each. This batching is vital for the efficient training of deep learning models, particularly when working with extensive datasets.
- In time series analysis, capturing temporal dependencies is paramount. The Time Series Generator is designed to maintain the chronological sequence of the data, providing the model with the context needed to learn from temporal relationships across time steps.

- Finally, the generator is integrated into the training workflow, supplying the model with consistent batches of input sequences alongside their corresponding labels for each training epoch, thereby streamlining the learning process.

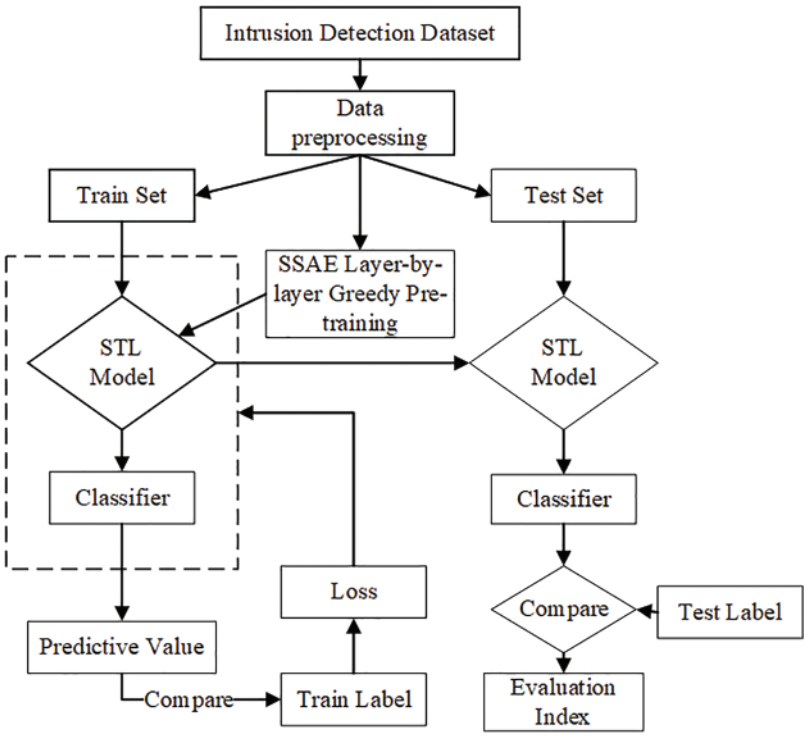


Figure 5: Experimental flow chart

5 Experiment Design and Comparative Analysis

5.1 Dataset

The experimental setup for this study is as follows: Intel(R) Xeon(R) CPU E5-2695 v3, 12 GB RAM, 50 GB NVMe, and a GeForce RTX 4070 Graphics Card. We use Python 3.8 and TensorFlow 2.11.0. The experiment uses the UNSW-NB15 and CICDS2017 datasets for intrusion detection.

The UNSW-NB15 dataset was proposed in 2015 by a research group from the Australian Security Center. This dataset is generated by simulating real network environments, making it more relevant to current network traffic [43]. The dataset is composed of 175,341 entries in its training segment and 82,332 in its testing counterpart. It encompasses nine distinct attack classifications: Generic, Exploit, Fuzzers, Denial of Service (DoS), Reconnaissance, Analysis, Backdoor, Shellcode, and Worms. From the dataset's 49 features, they can be categorized into 5 traffic attributes, 13 foundational attributes, 8 content attributes, 9 temporal attributes, 12 overarching attributes, and 2 label attributes. A detailed breakdown of the data count and distribution across the nine attack categories is depicted in Table 2.

Table 2: Various types of data in the UNSW-NB15 data set and their quantity distribution

Attack type	Train set		Test set	
	Quantity	Proportion/%	Quantity	Proportion/%
Normal	56000	31.94	37000	44.94
Generic	40000	22.81	18871	22.92
Exploit	33393	19.04	11132	13.52
Fuzzers	18184	10.37	6062	7.36
DoS	12264	6.99	4089	4.97
Reconnaissance	10491	5.98	3496	4.25
Analysis	2000	1.14	677	0.82
Backdoor	1746	1.00	583	0.71
Shellcode	1133	0.65	378	0.46
Worms	130	0.07	44	0.05
Total	175341	100.00	82332	100.00

CICIDS2017 is a reliable dataset proposed by Sharafaldin and others. It includes normal behaviors and 14 typical types of intrusion network data flows [44]. Conforming to real network scenarios, the dataset uses script files to simulate the normal behaviors of users in real scenarios. Additionally, during an attack, the mixture of normal user traffic data can better simulate real network intrusion scenarios. CICIDS2017 has a rich variety of attack samples. Compared to other open-source network intrusion detection datasets, it reflects well the current level of network attack techniques. Moreover, its network data feature information is abundant. The CICIDS2017 is a renowned dataset acknowledged by experts within the intrusion detection domain.

This dataset's data collection spanned five days, beginning at 9 a.m. on Monday, July 03, 2017, and concluding at 5 p.m. on July 07. The initial day, July 03, witnessed only regular network activities without any intrusions. The subsequent days recorded distinct attack patterns. Specifically, July 04 observed FTP-Patator and SSH-Patator brute force attacks alongside usual activities. July 05 was characterized by Denial of Service attacks with instances from Golden Eye, Hulk, Slow Http Test, Slow Loris, and Heartbleed. The next day, July 06, documented Web attacks such as XSS, SQL injection, Brute Force, and penetration assaults. Lastly, July 07 registered botnet invasions, port scanning, and Distributed Denial of Service attacks, notably DDoS LOIT. The attacks interspersed with regular activities simulated real-life network intrusions effectively. Details of the attacks and their quantities can be found in Table 3.

Table 3: Various types of data in the CICDS2017 data set and their quantity distribution

Attack type	Quantity
Normal	2271318
DDoS	128025
Port Scan	158804
Bot	1956

(Continued)

Table 3 (continued)

Attack type	Quantity
Infiltratior	36
Brute Force	1507
Sql Injection	21
XSS	652
FTP-Patator	7935
SSH-Patator	5897
Golden Eye	10293
Hulk	230124
Slow Http Test	5499
Slow Loris	5796
Heart Bleed	11

5.2 Data Preprocessing

The UNSW-NB15 dataset comprises both numerical and categorical features. As deep learning models necessitate numerical inputs, we transform the categorical attributes—"proto", "service", and "state"—into numerical counterparts via one-hot encoding. Specifically, the "proto" feature encompasses 133 unique symbolic values, "service" holds 13, and "state" includes 11. Consequently, post-transformation, the feature dimensionality escalates from 42 to 196. The dataset contains data of different dimensions that can be discrete or continuous, and they have a significant range difference, which makes values across different dimensions incomparable. To mitigate the impact of this significant range difference on model computations, the Min-Max normalization method is used to map the data samples into the [0,1] range. The preprocessed data is then fed into a stacked sparse autoencoder for dimensionality reduction.

The CICDS2017 dataset contains collected traffic data and does not have pre-divided training and test sets. Firstly, the data is cleaned, missing values are filled in, and erroneous content is corrected, converting "NaN" and "Infinite" values either to zero or replacing them with the average value. Subsequently, five irrelevant columns, such as IP addresses and timestamps, are removed from the dataset. Labels of 0 and 1 are then assigned to normal and attack traffic, respectively. Lastly, the data is transformed using one-hot encoding and Min-Max normalization, mapping all values into the [0,1] range. Min-Max normalization is represented by Eq. (13). The data preprocessing workflow is shown in Fig. 6.

$$x^* = \frac{x - min}{max - min} \tag{10}$$

Figure 6: Data preprocessing flow chart

5.3 Evaluation Indicator

The performance of the proposed intrusion detection model is assessed using a comprehensive set of evaluation metrics specifically chosen for their relevance in the context of security analytics. These include Accuracy, Precision, Recall, F1-score, and the False Alarm Rate (FAR), each providing unique insights into the model's detection capabilities [45].

Accuracy (Eq. (11)): Measures the proportion of both True Positives (TP) and True Negatives (TN) among all tested samples, giving a straightforward indication of overall correctness.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{11}$$

Precision (Eq. (12)): Evaluates the reliability of the model's positive predictions, highlighting its ability to minimize False Positives (FP), which is critical in preventing unnecessary responses to non-threats.

$$Precision = \frac{TP}{TP + FP} \tag{12}$$

Recall (Eq. (13)): Also known as the true positive rate, it assesses the model's ability to correctly identify all actual attacks, a key performance aspect in ensuring no threats go undetected.

$$Recall = \frac{TP}{TP + FN} \tag{13}$$

F1-score (Eq. (14)): The harmonic means of Precision and Recall serves as a single metric that balances both the false positives and false negatives, providing a holistic view of the model's precision and robustness in detection.

$$F1 - sorce = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{14}$$

False Alarm Rate (FAR) (Eq. (15)): Indicates the likelihood of the model incorrectly labeling normal traffic as malicious, which is crucial for maintaining user trust and operational efficiency.

$$FAR = \frac{FP}{TN + FP} \tag{15}$$

The selection of these metrics is grounded in the need to present a nuanced view of the model's performance, considering not only its effectiveness in identifying threats but also its precision and operational efficiency. This balanced approach to evaluation is particularly important in intrusion detection systems where the cost of misclassification can be high, both in terms of security breaches (when attacks are missed) and disrupted legitimate activities (when normal actions are falsely flagged as attacks).

5.4 Experimental Result

The model's training was bifurcated into two phases. In the inaugural phase, a greedy layer-wise pre-training approach was adopted to ascertain the optimal number of layers for the SSAE. Initially, the structure of a single-layer sparse autoencoder was defined. Subsequently, layer-wise pre-training was employed to train SSAE with varying layers. This was then followed by connecting it to a single-layer LSTM network. The performance was tested with a time step length of 1, ultimately saving the most optimal SSAE model. The test results are shown in Table 4. Based on the definition of the sparse

autoencoder, it is evident that its performance is primarily determined by factors such as the number of units, hidden layers, and sparsity constant.

Table 4: UNSW-NB15 data set SSAE parameter selection

SSAE	ρ	Acc (train)	Acc (test)	FAR (train)	FAR (test)
[64,32,32]	0.02	0.9468	0.8867	0.0851	0.2381
	0.04	0.9441	0.8870	0.0780	0.2208
	0.06	0.9438	0.8826	0.0810	0.2294
[64,64,32]	0.02	0.9348	0.8868	0.0602	0.1840
	0.04	0.9362	0.8920	0.0572	0.1688
	0.06	0.9447	0.8847	0.0785	0.2194
[128,64,32]	0.02	0.9403	0.9057	0.0620	0.1804
	0.04	0.9457	0.8922	0.0775	0.2213
	0.06	0.9458	0.0792	0.8844	0.2272
[128,32,32]	0.02	0.9364	0.8781	0.0810	0.2287
	0.04	**0.9489**	**0.8989**	**0.0524**	**0.1620**
	0.06	0.9387	0.8888	0.0635	0.1838

Analyzing the preprocessed 196-dimensional data from the UNSW-NB15 dataset, Table 4 reveals that the SSAE [128,32,32] equipped with a sparsity constant $\rho = 0.04$, stands out with the pinnacle of training and testing accuracies and the nadir of FAR, hence it was selected for data dimensionality reduction. The CICDS2017 dataset lacks categorical features and after preprocessing, it only has 79-dimensional data. If we continue with the layer design of UNSW-NB15, the data would first be elevated from 79 dimensions to 128, making it sparser and unable to extract key features in subsequent stages. Scrutinizing the figures in Table 5, the SSAE [64,32,32] with sparsity constant $\rho = 0.04$ emerges as the frontrunner in accuracy and the benchmark for the lowest FAR. Consequently, empirical evidence steered the parameter determinations for the SSAE across both datasets.

Table 5: CICDS2017 data set SSAE parameter selection

SSAE	ρ	Acc (train)	Acc (test)	FAR (train)	FAR (test)
[128,64,32]	0.02	0.9860	0.9861	0.0097	0.0099
	0.04	0.9809	0.9808	0.0167	0.0169
	0.06	0.9823	0.9823	0.0173	0.0176
[128,32,32]	0.02	0.9837	0.9839	0.0139	0.0140
	0.04	0.9840	0.9839	0.0125	0.0128
	0.06	0.9826	0.9825	0.0151	0.0153

(Continued)

Table 5 (continued)

SSAE	ρ	Acc (train)	Acc (test)	FAR (train)	FAR (test)
[32,32,32]	0.02	0.9802	0.9803	0.0169	0.0170
	0.04	0.9814	0.9813	0.0169	0.0171
	0.06	0.9836	0.9837	0.0148	0.0148
[64,32,32]	0.02	0.9825	0.9825	0.0138	0.0140
	0.04	**0.9870**	**0.9871**	**0.0086**	**0.0087**
	0.06	0.9827	0.9823	0.0089	0.0088

In the subsequent phase, data processed by SSAE serves as the foundation for the time-series model TCN-BiLSTM. Prolonged experimentation involving both TCN and Bi-LSTM yielded our chosen TCN parameters as follows: nb_filters = 64, kernel_size = 2, nb_stacks = 1, and dilations = [1, 2, 4, 8, 16]. For the Bi-LSTM, a neuron structure of [24,12] was adopted. To bolster the model's resilience and impede overfitting, a Dropout layer with a rate of 0.4 was integrated post the assembly of TCN and Bi-LSTM networks.

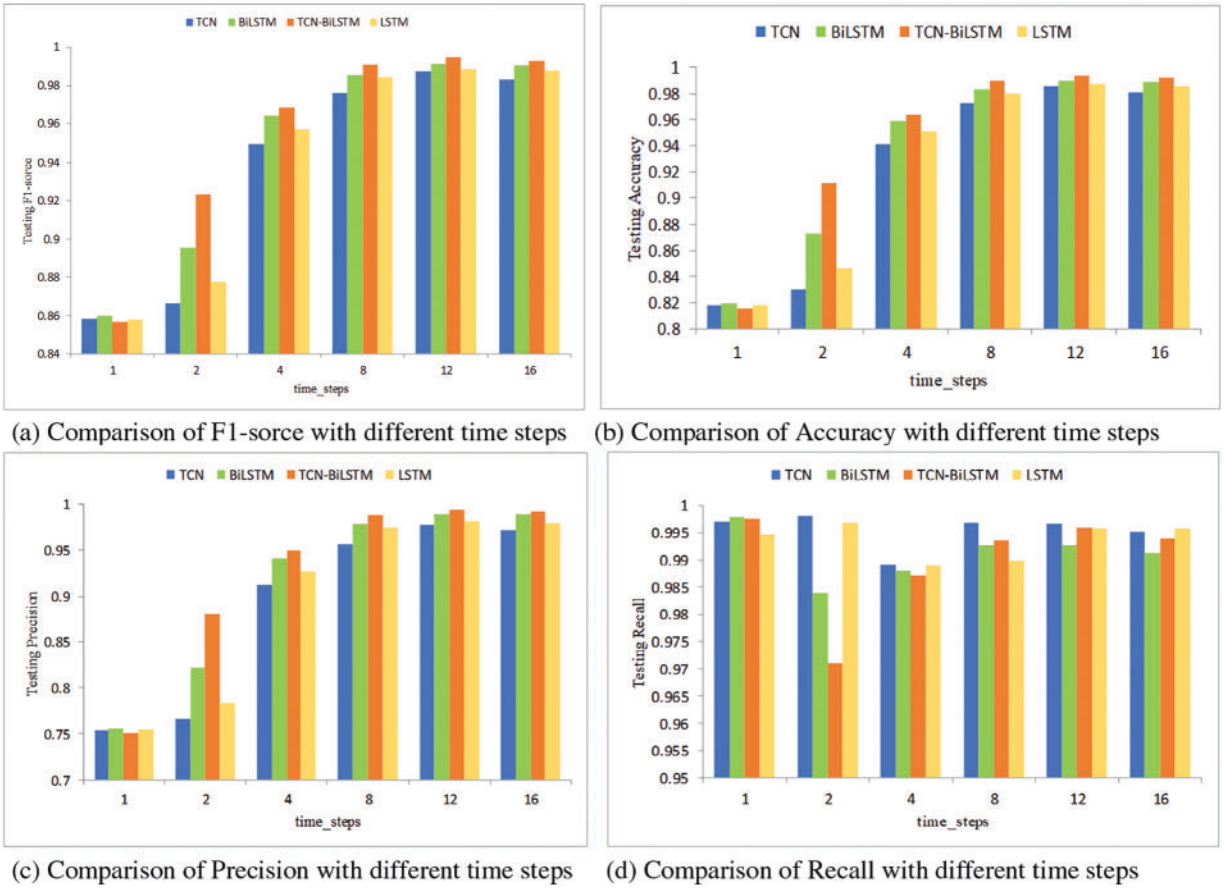
Table 6 presents a comparative analysis of the effects of SSAE on the dimensionality reduction for two datasets, specifically highlighting the results at time steps 2 and 4, where a notable increase in the STL hybrid model's performance is observed. The data from Table 5 demonstrates that, at these specific time steps, the performance metrics for both datasets, post-application of SSAE for dimensionality reduction, are significantly higher compared to those metrics without dimensionality reduction. This finding leads to a clear conclusion: employing SSAE for data dimensionality reduction effectively enhances the STL hybrid model's performance.

Table 6: SSAE ablation experiment

Dataset	SSAE	Time steps	F1-sorce	Accuracy	Precision	Recall
UNSW-NB15	No	2	0.8956	0.8730	0.8218	0.9840
		4	0.9565	0.9503	0.9220	0.9937
	Yes	2	0.9236	0.9110	0.8805	0.9711
		4	0.9683	0.9642	0.9501	0.9871
CICDS2017	No	2	0.9550	0.9489	0.9266	0.9851
		4	0.9645	0.9858	0.9518	0.9773
	Yes	2	0.9764	0.9903	0.9565	0.9969
		4	0.9913	0.9947	0.9875	0.9952

The Timeseries Generator was harnessed to simulate the repercussions of historical network activities on intrusion detection. This was instrumental in corroborating the model's adeptness with time series data. An extensive time step might overlook quintessential events or patterns that transpire momentarily. Conversely, an overly minute time step can saturate the data with noise, complicating the model and possibly instigating overfitting. An analysis of Fig. 7 with the UNSW-NB15 dataset reveals

an ascent in detection accuracy across four models as the time step amplifies. This ascent plateaus at a time step of 12, with a subsequent time step of 16 manifesting a slight ebb in detection proficiency. It underscores the optimality of our proposed hybrid model at a time step of 12, boasting metrics like F1-score, Accuracy, Precision, and Recall at 99.49%, 99.43%, 99.38%, and 99.60%, respectively.



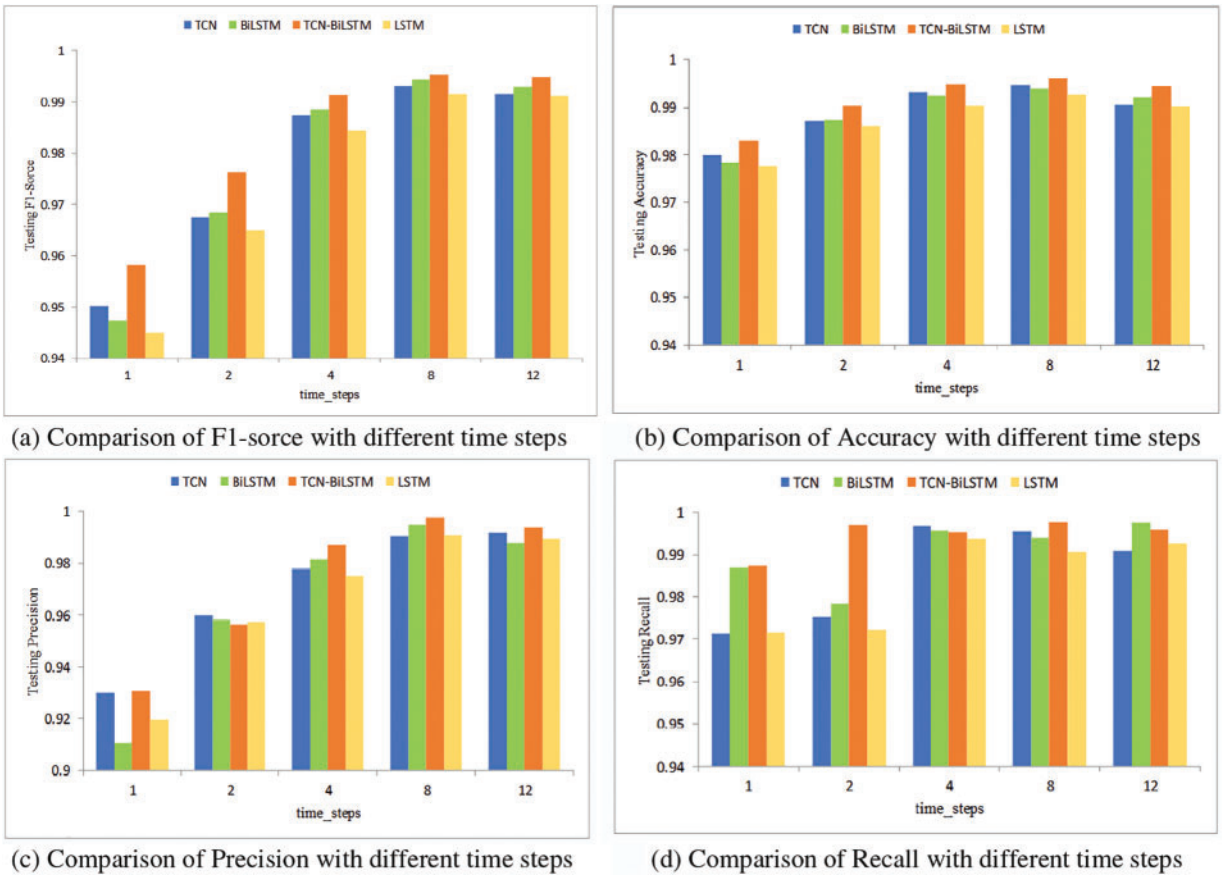
(a) Comparison of F1-sorce with different time steps (b) Comparison of Accuracy with different time steps

(c) Comparison of Precision with different time steps (d) Comparison of Recall with different time steps

Figure 7: UNSW-NB15 dataset experimental results

Fig. 8 showcases that, for the CICDS2017 dataset, our proposed hybrid model reaches its zenith of performance at a time step of 8, delivering impressive F1-score, accuracy, precision, and recall values of 99.53%, 99.62%, 99.27%, and 99.79%, respectively. This signifies the commendable generalization abilities of the SSAE-TB hybrid model.

Fig. 9 depicts a comparative line chart illustrating the training durations of the hybrid model on two distinct datasets. This experiment was conducted using a batch size of 1024 over 250 epochs. The chart clearly shows that dimensionality reduction via SSAE leads to decreased training times, with simpler model combinations benefiting the most. The STL model, which integrates three deep learning approaches, unsurprisingly necessitates longer training periods. Furthermore, utilizing data that has not undergone SSAE's dimensionality reduction in the hybrid model results in increased durations for feature extraction and attack detection. Consequently, employing SSAE to reduce data dimensionality before the training phase with the TCN and Bi-LSTM networks is both a logical and practical approach.



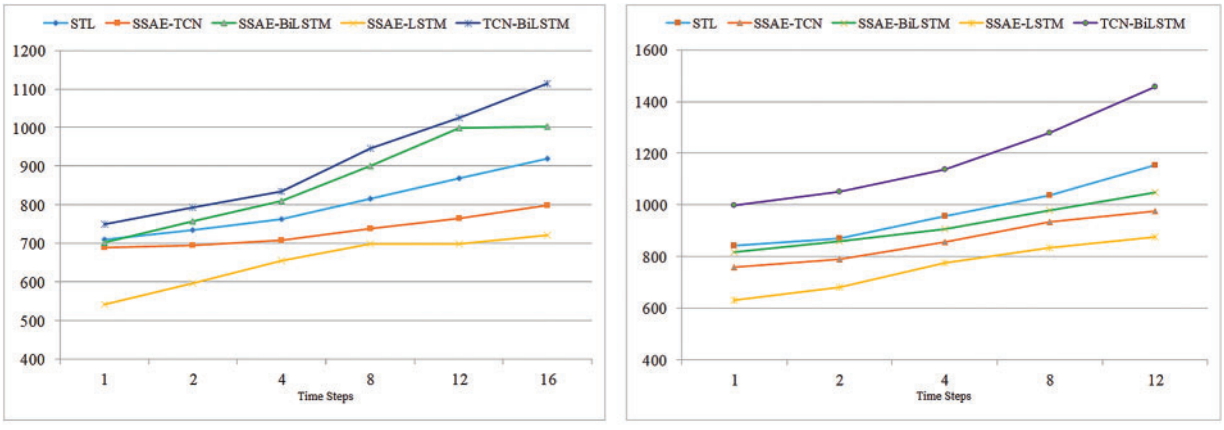
(a) Comparison of F1-sorce with different time steps

(b) Comparison of Accuracy with different time steps

(c) Comparison of Precision with different time steps

(d) Comparison of Recall with different time steps

Figure 8: CICDS2017 dataset experimental results



(a) UNSW-NB15 dataset

(b) CICDS2017 dataset

Figure 9: Training times comparison

Fig. 10 presents a line graph that compares the inference times of a hybrid model utilizing two different datasets. Consistent with the training phase, data that has been subjected to SSAE dimensionality reduction before being processed by deep learning methods show a marked improvement in detection results over non-reduced data. While the inference times for various hybrid models post-SSAE reduction are comparable, the STL hybrid model distinguishes itself by achieving higher accuracy and F1-scores at different temporal steps. In the context of network intrusion detection, the precision of attack detection is paramount for enacting prompt countermeasures. Despite a marginal difference in inference time—mere seconds compared to other models—the STL hybrid model's superior performance metrics solidify its advantage.

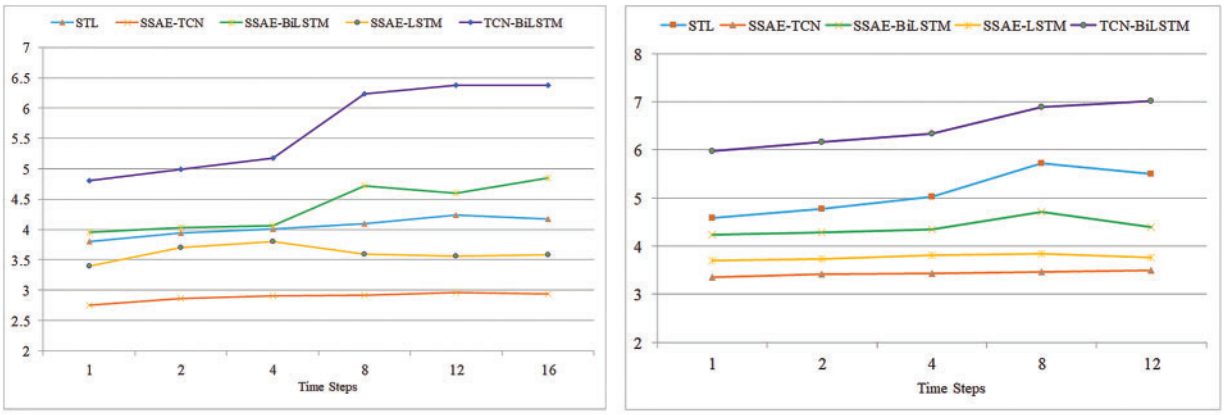


Figure 10: Inference times comparison

The performance of our SSAE-TB hybrid model was juxtaposed against prevailing advanced models using the UNSW-NB15 and CICDS2017 datasets, as delineated in Tables 7 and 8. Key metrics like F1-score, accuracy, precision, and recall were brought into consideration for comparison. Analyzing Table 7 reveals that the STL model clinches the top spot for F1-score and accuracy on the UNSW-NB15 dataset, with figures standing at 0.9949 and 0.9943, respectively. In the context of the CICDS2017 dataset, while its accuracy trails slightly behind the LSTM-DNN method, the model surpasses the other three evaluation criteria.

Table 7: Comparative experiment of the model on the UNSW-NB15 data set

Studies	Year	Method	F1-sorce	Accuracy	Precision	Recall
Udas et al. [46]	2022	SPIDER	0.8612	0.8246	0.7630	0.9885
Zheng et al. [47]	2019	GCN-TC	0.9689	0.9697	0.9695	0.9687
Xu et al. [32]	2022	LogAE-XGBoost	0.9645	0.9511	0.9549	0.9645
Han et al. [48]	2022	IDHCS	0.9427	0.9451	0.8975	0.9927
Thakkar et al. [49]	2023	FST-DNN	0.9693	0.8903	0.9500	0.9895
Ozkan-Okay [50]	2023	FSACM	0.9713	0.9884	0.9727	0.9699
Kumar et al. [51]	2023	NBGOA	0.9696	0.9917	0.9523	0.9876
Imran et al. [33]	2021	AutoML-kalman filter	0.9876	0.9880	0.9876	0.9877
Wang et al. [34]	2023	RUIDS	0.9544	0.9262	0.9489	0.9601

(Continued)

Table 7 (continued)

Studies	Year	Method	F1-sorce	Accuracy	Precision	Recall
Deng et al. [38]	2022	FT-GCN	0.9865	0.9872	0.9866	0.9865
Proposed model	2023	STL	**0.9949**	**0.9943**	**0.9938**	**0.9960**

Table 8: Comparative experiment of the model on the CICDS2017 data set

Studies	Year	Method	F1-sorce	Accuracy	Precision	Recall
Imran et al. [33]	2021	AutoML-kalman filter	0.9603	0.9702	0.9485	0.9724
Han et al. [48]	2022	IDHCS	0.9655	0.9655	0.9660	0.9650
Toldinas et al. [52]	2021	MDLIR	0.9848	0.9849	0.9849	0.9848
Liu et al. [53]	2019	FFT-CNN	0.9869	0.9870	0.9870	0.9869
Gu et al. [54]	2021	NB-SVM	0.9821	0.9892	0.9946	0.9700
Hammad et al. [55]	2021	T-SNERF	0.9890	0.9878	0.9890	0.9890
Yang et al. [56]	2021	MTH-IDS	0.9889	0.9889	0.9890	0.9886
Lopez-Martin et al. [35]	2021	RBFNN	0.9946	0.9952	0.9935	0.9957
Kim et al. [36]	2022	LSTM-DNN	0.9896	0.9974	0.9896	0.9897
Siddiqi et al. [37]	2022	Gabor filter-CNN	0.9877	0.9879	0.9880	0.9877
Proposed model	2023	STL	**0.9953**	**0.9962**	**0.9927**	**0.9979**

Table 9 details the training and inference times for a range of deep learning models, encompassing studies from comparative experiments conducted on the two previously mentioned datasets. In practical scenarios of network intrusion detection, the detection system—once trained—operates in real-time; thus, longer training times do not impede the system's operational performance. Nevertheless, the inference time is of paramount importance as it directly reflects the model's ability to quickly identify attacks, an essential feature for the timely detection and mitigation of potential network threats.

To validate the superiority of the STL model presented in this paper over the other studies discussed, we conducted a comprehensive analysis using key performance indicators such as accuracy, F1-score, and inference time.

- The STL model demonstrated remarkable results, achieving an accuracy of 99.43% and an F1-score of 99.49% on the UNSW-NB15 dataset; and even higher scores of 99.62% and 99.53%, respectively, on the CICIDS2017 dataset. These figures not only exceed those of many competing models but also underscore the STL model's robustness in detecting network intrusions.
- In addition, the STL model proved to be significantly faster, recording inference times of just 4.24 and 5.72 s. This performance is at least five times quicker than its counterparts, marking a critical advantage in real-time threat detection scenarios.

Table 9: Time comparison of different models

Studies	Method	Dataset	Training time	Inference time
Thakkar et al. [49]	FST-DNN	UNSW-NB15	13913.5 s	*
Yang et al. [56]	MTH-IDS	CICDS2017	478.2 s	*
Xu et al. [32]	LogAE-XGBoost	UNSW-NB15	132 s	*
Kumar et al. [51]	NBGOA	UNSW-NB15	47 s	23 s
Hnamte et al. [57]	LSTM-AE	CICDS2017	184 s	53.66 s
Hnamte et al. [58]	DNN	CICDS2017	33 s	29.05 s
	DCNN		40 s	29.36 s
Proposed model	STL	UNSW-NB15	869.23 s	4.24 s
		CICDS2017	1031.75 s	5.72 s

The combination of high detection rates and rapid inference underlines the STL model's competitive edge in the field of network intrusion detection. In conclusion, the STL model emerges as an exceptional solution, providing high accuracy, superior F1-scores, and swift inference times, and thereby outperforming the models in comparative studies.

6 Conclusion

In this research, we have introduced the STL hybrid model, meticulously crafted to tackle the complexity of network intrusion detection. Our innovative approach bridges the gap in conventional methods that often neglect the critical time-series dynamics, thus enhancing detection accuracy while reducing false positives and negatives. Through the application of an SSAE for feature reduction and a layered pre-training methodology, we have distilled an optimal configuration for our model. A time series generator has further enriched the model to reflect the intricacies of real-world data flow. The fusion of TCN and Bi-LSTM in our hybrid model stands as a testament to our commitment to precision, with the TCN adeptly unraveling the multivariate time series data and the Bi-LSTM refining the sequence processing. The subsequent experimental phase, marked by variations in time step length, has solidified the prowess of our hybrid model in surmounting the challenges inherent to time series data.

Impact and Implications: The outcomes of our research have significant implications for the field of intrusion detection. By successfully capturing the time-dependent nature of network traffic, our STL model paves the way for more nuanced and responsive intrusion detection systems. It is not merely a theoretical enhancement but a practical leap forward in safeguarding network infrastructures.

Challenges and Limitations: It is pertinent to acknowledge the challenges encountered during our research. We grappled with computational constraints and the intricate nature of crafting a model that could generalize across diverse network environments. Our model, while robust, may face limitations in scenarios of extreme data volatility or in instances where attack patterns drastically deviate from historical trends.

Future Directions: As we look to the horizon, we are compelled to delve into the untapped potential of real-time data collection. Our ambition is to construct comprehensive datasets that will allow for an in-depth examination of the variable impacts of time series on network intrusion detection.

We seek to refine our model to achieve even greater speed and accuracy, thus contributing to the ever-evolving landscape of cybersecurity.

Call to Action: We call upon the research community to build upon our findings, explore new dimensions of time series analysis, and push the boundaries of what is possible in network intrusion detection. The STL model is a stepping stone toward a future where digital fortifications are as dynamic and intricate as the networks they protect.

Acknowledgement: The author thanks his family and colleague for their moral support.

Funding Statement: This work was supported in part by the Gansu Province Higher Education Institutions Industrial Support Program: Security Situational Awareness with Artificial Intelligence and Blockchain Technology. Project Number (2020C-29).

Author Contributions: For Conceptualization, Methodology, Validation, Writing, and Software, X.W.; Conceptualization, C.L.; Supervision and Funding Acquisition, Z.H. All authors have read and agreed to the published version of the manuscript.

Availability of Data and Materials: The data used in this study are not publicly available due to ongoing research projects that utilize the same dataset. The data will be considered for release after the completion of these projects.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] L. Ashiku and C. Dagli, "Network intrusion detection system using deep learning," *Procedia Computer Science*, vol. 185, pp. 239–247, 2021.

[2] J. Snehi, A. Bhandari and G. Singh, "Review of existing data sets for network intrusion detection system," *Advances in Mathematics: Scientific Journal*, vol. 9, pp. 3849–3854, 2020. https://doi.org/10.37418/amsj.9.6.64

[3] J. Snehi, M. Snehi, A. Bhandari, V. Baggan and R. Ahuja, "Introspecting intrusion detection systems in dealing with security concerns in cloud environment," in *2021 10th Int. Conf. on System Modeling & Advancement in Research Trends (SMART)*, Moradabad, India, IEEE, pp. 345–349, 2021.

[4] J. Snehi, A. Bhandari, V. Baggan, M. Snehi and H. Kaur, "AIDAAS: Incident handling and remediation anomaly-based IDaaS for cloud service providers," in *2021 10th Int. Conf. on System Modeling & Advancement in Research Trends (SMART)*, Moradabad, India, pp. 356–360, 2021. https://doi.org/10.1109/SMART52563.2021.9676296

[5] J. Verma, A. Bhandari and G. Singh, "iNIDS: SWOT analysis and TOWS inferences of state-of-the-art NIDS solutions for the development of intelligent network intrusion detection system," *Computer Communications*, vol. 195, pp. 227–247, 2022. https://doi.org/10.1016/j.comcom.2022.08.022

[6] J. Verma, A. Bhandari and G. Singh, "A meta-analysis of role of network intrusion detection systems in confronting network attacks," in *2021 8th Int. Conf. on Computing for Sustainable Global Development (INDIACom)*, New Delhi, India, pp. 506–511, 2021.

[7] P. Sun, P. Liu, Q. Li, C. Liu, X. Lu *et al.,* "DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system," *Security and Communication Networks*, vol. 2020, pp. 1–11, 2020.

[8] J. Snehi, A. Bhandari, V. Baggan, M. Snehi and Ritu, "Diverse methods for signature based intrusion detection schemes adopted," *International Journal of Recent Technology and Engineering*, vol. 9, no. 2, pp. 44–49, 2020. https://doi.org/10.35940/ijrte.A2791.079220

[9] A. Khraisat, I. Gondal, P. Vamplew and J. Kamruzzaman, "Survey of intrusion detection systems: Techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.
[10] J. Snehi, A. Bhandari, M. Snehi, U. Tandon and V. Baggan, "Global intrusion detection environments and platform for anomaly-based intrusion detection systems," in *Proc. of Second Int. Conf. on Computing, Communications, and Cyber-Security: IC4S 2020*, Ghaziabad, India, Springer, pp. 817–831, 2021.
[11] M. A. Siddiqi and N. Ghani, "Critical analysis on advanced persistent threats," *International Journal of Computer Applications*, vol. 141, no. 13, pp. 46–50, 2016.
[12] E. M. Kornaropoulos, C. Papamanthou and R. Tamassia, "The state of the uniform: Attacks on encrypted databases beyond the uniform query distribution," in *2020 IEEE Symp. on Security and Privacy (SP)*, San Francisco, California, USA, IEEE, pp. 1223–1240, 2020.
[13] J. Verma, A. Bhandari and G. Singh, "Feature selection algorithm characterization for NIDS using machine and deep learning," in *2022 IEEE Int. IOT, Electronics and Mechatronics Conf. (IEMTRONICS)*, Toronto, Canada, IEEE, pp. 1–7, 2022. https://doi.org/10.1109/IEMTRONICS55184.2022.9795709
[14] D. Chamou, P. Toupas, E. Ketzaki, S. Papadopoulos, K. Giannoutakis *et al.,* "Intrusion detection system based on network traffic using deep neural networks," in *2019 IEEE 24th Int. Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, Limassol, Cyprus, IEEE, pp. 1–6, 2019. https://doi.org/10.1109/CAMAD.2019.8858475
[15] F. Wen, S. Tao and L. Jie, "Overview of deep learning principles and applications," *Computer Science*, vol. 45, pp. 11–15+40, 2018.
[16] W. Wang, M. Zhu, J. Wang, X. Zeng and Z. Yang, "End-to-end encrypted traffic classification with one-dimensional convolution neural networks," in *2017 IEEE Int. Conf. on Intelligence and Security Informatics (ISI)*, Beijing, China, IEEE, pp. 43–48, 2017. https://doi.org/10.1109/ISI.2017.8004872
[17] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink and J. Schmidhuber, "LSTM: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 10, pp. 2222–2232, 2017. https://doi.org/10.1109/TNNLS.2016.2582924
[18] B. Plank, A. Søgaard and Y. Goldberg, "Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss," arXiv preprint arXiv:1604.05529, 2016.
[19] S. Bai, J. Z. Kolter and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," arXiv preprint arXiv:1803.01271, 2018.
[20] J. P. Anderson, "Computer security threat monitoring and surveillance," *Technical Report*, James P. Anderson Company, 1980.
[21] H. Debar, M. Becker and D. Siboni, "A neural network component for an intrusion detection system," in *IEEE Symp. on Security and Privacy*, Oakland, CA, USA, pp. 240–250, 1992. https://doi.org/10.1109/ISI.2017.8004872
[22] W. H. Lin, H. C. Lin, P. Wang, B. H. Wu and J. Y. Tsai, "Using convolutional neural networks to network intrusion detection for cyber threats," in *2018 IEEE Int. Conf. on Applied System Invention (ICASI)*, Chiba, Japan, IEEE, pp. 1107–1110, 2018. https://doi.org/10.1109/ISI.2017.8004872
[23] R. Vinayakumar, K. Soman and P. Poornachandran, "A comparative analysis of deep learning approaches for network intrusion detection systems (N-IDSs): Deep learning for N-IDSs," *International Journal of Digital Crime and Forensics (IJDCF)*, vol. 11, no. 3, pp. 65–89, 2019.
[24] G. C. Fernández and S. Xu, "A case study on using deep learning for network intrusion detection," in *MILCOM 2019-2019 IEEE Military Communications Conf. (MILCOM)*, Norfolk, VA, USA, IEEE, pp. 1–6, 2019. https://doi.org/10.1109/ISI.2017.8004872
[25] B. J. Radford, L. M. Apolonio, A. J. Trias and J. A. Simpson, "Network traffic anomaly detection using recurrent neural networks," arXiv preprint arXiv:1803.10769, 2018.
[26] S. P. Thirimanne, L. Jayawardana, L. Yasakethu, P. Liyanaarachchi and C. Hewage, "Deep neural network based real-time intrusion detection system," *SN Computer Science*, vol. 3, no. 2, pp. 145, 2022.
[27] T. Pooja and P. Shrinivasacharya, "Evaluating neural networks using bi-directional LSTM for network IDS (intrusion detection systems) in cyber security," *Global Transitions Proceedings*, vol. 2, no. 2, pp. 448–454, 2021.

[28] T. Thilagam and R. Aruna, "Intrusion detection for network based cloud computing by custom RC-NN and optimization," *ICT Express*, vol. 7, no. 4, pp. 512–520, 2021.
[29] H. Wang, J. Liu, J. Kang, W. Yin, H. Sun *et al.,* "Feature envy detection based on Bi-LSTM with self-attention mechanism," in *2020 IEEE Int. Conf. on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, Exeter, UK, IEEE, pp. 448–457, 2020. https://doi.org/10.1109/ISI.2017.8004872
[30] J. Sinha and M. Manollas, "Efficient deep CNN-BiLSTM model for network intrusion detection," in *Proc. of the 2020 3rd Int. Conf. on Artificial Intelligence and Pattern Recognition*, Xiamen, China, pp. 448–457, 2020. https://doi.org/10.1109/ISI.2017.8004872
[31] G. Xu, L. Xiao, J. Ru and G. Yu, "Intrusion detection based on improved sparse denoising autoencoder," *Journal of Computer Applications*, vol. 39, pp. 769–773, 2019.
[32] W. Xu and Y. Fan, "Intrusion detection systems based on logarithmic autoencoder and XGBoost," *Security and Communication Networks*, vol. 2022, pp. 9068724, 2022. https://doi.org/10.1155/2022/9068724
[33] F. J. Imran and D. Kim, "An ensemble of prediction and learning mechanism for improving accuracy of anomaly detection in network intrusion environments," *Sustainability*, vol. 13, no. 18, pp. 10057, 2021.
[34] W. Wang, S. Jian, Y. Tan, Q. Wu and C. Huang, "Robust unsupervised network intrusion detection with self-supervised masked context reconstruction," *Computers & Security*, vol. 128, pp. 103131, 2023. https://doi.org/10.1016/j.cose.2023.103131
[35] M. Lopez-Martin, A. Sanchez-Esguevillas, J. I. Arribas and B. Carro, "Network intrusion detection based on extended RBF neural network with offline reinforcement learning," *IEEE Access*, vol. 9, pp. 153153–153170, 2021. https://doi.org/10.1109/ACCESS.2021.3127689
[36] T. Kim and W. Pak, "Early detection of network intrusions using a GAN-based one-class classifier," *IEEE Access*, vol. 10, pp. 119357–119367, 2022.
[37] M. A. Siddiqi and W. Pak, "Tier-based optimization for synthesized network intrusion detection system," *IEEE Access*, vol. 10, pp. 108530–108544, 2022.
[38] X. Deng, J. Zhu, X. Pei, L. Zhang, Z. Ling *et al.,* "Flow topology-based graph convolutional network for intrusion detection in label-limited IoT networks," *IEEE Transactions on Network and Service Management*, vol. 20, no. 1, pp. 684–696, 2022.
[39] M. E. Karim, M. M. S. Maswood, S. Das and A. G. Alharbi, "BHyPreC: A novel Bi-LSTM based hybrid recurrent neural network model to predict the CPU workload of cloud virtual machine," *IEEE Access*, vol. 9, pp. 131476–131495, 2021. https://doi.org/10.1109/ACCESS.2021.3113714
[40] Z. Chun, "Research and application of sparse deep models based on autoencoders," *Journal of Xidian University*, vol. 44, pp. 36–42, 2017 (In Chinese). https://doi.org/10.3969/j.issn.1001-2400.2017.03.007
[41] J. Tang and Y. R. Chien, "Research on wind power short-term forecasting method based on temporal convolutional neural network and variational modal decomposition," *Sensors*, vol. 22, no. 19, pp. 7414, 2022.
[42] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.
[43] N. Moustafa and J. Slay, "UNSW-NB15: A comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set)," in *2015 Military Communications and Information Systems Conf. (MilCIS)*, Canberra, Australia, IEEE, pp. 1–6, 2015. https://doi.org/10.1109/ISI.2017.8004872
[44] I. Sharafaldin, A. H. Lashkari and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *ICISSp*, vol. 1, pp. 108–116, 2018.
[45] D. M. Powers, "Evaluation: From precision, recall and F-measure to ROC, informedness, markedness and correlation," arXiv preprint arXiv:2010.16061, 2010.
[46] P. B. Udas, M. E. Karim and K. S. Roy, "SPIDER: A shallow PCA based network intrusion detection system with enhanced recurrent neural networks," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 10, pp. 10246–10272, 2022.

[47] J. Zheng and D. Li, "GCN-TC: Combining trace graph with statistical features for network traffic classification," in *ICC 2019-2019 IEEE Int. Conf. on Communications (ICC)*, Shanghai, China, IEEE, pp. 1–6, 2019. https://doi.org/10.1109/ISI.2017.8004872
[48] H. Han, H. Kim and Y. Kim, "An efficient hyperparameter control method for a network intrusion detection system based on proximal policy optimization," *Symmetry*, vol. 14, no. 1, pp. 161, 2022.
[49] A. Thakkar and R. Lohiya, "Fusion of statistical importance for feature selection in deep neural network-based intrusion detection system," *Information Fusion*, vol. 90, pp. 353–363, 2023. https://doi.org/10.1016/j.inffus.2022.09.026
[50] M. Ozkan-Okay, R. Samet, Ö. Aslan, S. Kosunalp, T. Iliev *et al.,* "A novel feature selection approach to classify intrusion attacks in network communications," *Applied Sciences*, vol. 13, no. 19, pp. 11067, 2023.
[51] G. S. C. Kumar, R. K. Kumar, K. P. V. Kumar, N. R. Sai and M. Brahmaiah, "Deep residual convolutional neural network: An efficient technique for intrusion detection system," *Expert Systems with Applications*, vol. 238, pp. 121912, 2023. https://doi.org/10.1109/ACCESS.2023.3266979
[52] J. Toldinas, A. Venčkauskas, R. Damaševičius, Š. Grigaliūnas, N. Morkevičius *et al.,* "A novel approach for network intrusion detection using multistage deep learning image recognition," *Electronics*, vol. 10, no. 15, pp. 1854, 2021.
[53] W. Liu, X. Liu, X. Di and H. Qi, "A novel network intrusion detection algorithm based on fast fourier transformation," in *2019 1st Int. Conf. on Industrial Artificial Intelligence (IAI)*, Shenyang, China, IEEE, pp. 1–6, 2019. https://doi.org/10.1109/ISI.2017.8004872
[54] J. Gu and S. Lu, "An effective intrusion detection approach using SVM with naïve Bayes feature embedding," *Computers & Security*, vol. 103, pp. 102158, 2021.
[55] M. Hammad, N. Hewahi and W. Elmedany, "T-SNERF: A novel high accuracy machine learning approach for intrusion detection systems," *IET Information Security*, vol. 15, no. 2, pp. 178–190, 2021.
[56] L. Yang, A. Moubayed and A. Shami, "MTH-IDS: A multitiered hybrid intrusion detection system for internet of vehicles," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 616–632, 2021.
[57] V. Hnamte, H. Nhung-Nguyen, J. Hussain and Y. Hwa-Kim, "A novel two-stage deep learning model for network intrusion detection: LSTM-AE," *IEEE Access*, vol. 11, pp. 37131–37141, 2023. https://doi.org/10.1109/ACCESS.2023.3266979
[58] V. Hnamte and J. Hussain, "Dependable intrusion detection system using deep convolutional neural network: A novel framework and performance evaluation approach," *Telematics and Informatics Reports*, vol. 11, pp. 100077, 2023.