International Journal of Advanced Technology and Engineering Exploration, Vol 12(124)
ISSN (Print): 2394-5443 ISSN (Online): 2394-7454
http://dx.doi.org/10.19101/IJATEE.2024.111100055

Research Article

ResNet50-based deep convolutional neural network for zero-day attack prediction and detection

Swathy Akshaya[1*] and Padmavathi. G[2]
Research Scholar, Department of Computer Science, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, Tamil Nadu, India[1]
Professor, Department of Computer Science, Avinashilingam Institute for Home Science and Higher Education for Women, Coimbatore, Tamil Nadu, India[2]

Received: 13-January-2024; Revised: 24-February-2025; Accepted: 09-March-2025
©2025 Swathy Akshaya and Padmavathi. G. This is an open access article distributed under the Creative Commons Attribution (CC BY) License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Abstract
A zero-day attack (ZDA) is a cyberattack that targets networks and systems by exploiting previously unknown security vulnerabilities. Software vendors have zero days to identify, address, and patch newly discovered threats, hence the term "zero-day." In cybersecurity, effectively detecting and mitigating malicious nodes is crucial, particularly against zero-day malware. Traditional antivirus systems, which rely on stored malware signatures, struggle to detect ZDAs, making them vulnerable to advanced malware specifically designed to evade detection. To address this challenge, a novel approach called deep convolutional n-zero-day adversarial safety network (DC-nZDASN) has been proposed. This method trains a model to distinguish between real and synthetic malware samples by generating artificial malware data. The synthetic data introduces new characteristics that contrast with the original dataset, enhancing the model's detection capability. The proposed approach incorporates multiple malware features and utilizes real-world and network traffic datasets for model development. During preprocessing, the standard scaler is applied, and decision tree regression (DTR) is used, while feature selection is performed using random forest (RF) in combination with logistic regression (LR). The model is trained and tested using residual network (ResNet50), long short-term memory (LSTM), and convolutional neural network (CNN). For classification, various machine learning (ML) algorithms, such as decision tree (DT), LR, support vector machine (SVM), gaussian naïve bayes (GNB), and stacking ensemble classification (SEC), are employed. The proposed DC-nZDASN model achieves a classification accuracy of 95.09%, demonstrating a significant advancement in malware detection, particularly for zero-day threats. By leveraging generated synthetic malware samples, the model enhances its ability to detect novel threats, outperforming traditional methods. The integration of preprocessing techniques, feature selection, and a diverse set of ML algorithms further improves the model's overall effectiveness.

Keywords
Zero-day attack, Deep convolutional neural network (DCNN), Resnet50, Malware detection, Transfer learning, Machine learning.

1.Introduction
Information and communication technologies have accelerated human tasks while introducing new risks, such as network model intrusions [1]. Intrusion in information systems refers to unauthorized attacks on data integrity, availability, and confidentiality, including data alteration or destruction [2]. Predicting zero-day attacks (ZDA) remains one of the biggest challenges in intrusion detection systems (IDS) [3].

Figure 1 illustrates the zero-day vulnerability timeline architecture, outlining the lifecycle of security flaws from discovery to exploitation and eventual patching [4, 5].

To mitigate these risks, researchers have developed IDS solutions to identify and reduce harmful network activity, ensuring system security [6, 7]. Most IDS are either anomaly-based or signature-based [8]. Anomaly-based IDS can detect irregularities in normal network activity patterns, making them effective in identifying unknown attacks, though they tend to produce more false positives (FPs) [9]. In contrast, signature-based IDS are effective at

*Author for correspondence

recognizing known attack patterns but cannot detect new, previously unrecorded threats [10].

Figure 1 Zero-day vulnerability timeline

ZDA prediction exploits undisclosed vulnerabilities in software, making it one of the most complex challenges in IDS [11]. ZDAs pose a significant threat as they target vulnerable systems before a fix is available [12]. *Figure 2* illustrates the lifecycle of a ZDA, detailing the stages from vulnerability discovery to exploitation. These attacks typically involve multiple steps, including vulnerability identification, weaponization, and exploitation [13]. Signature-based IDS are ineffective in detecting ZDAs, whereas anomaly-based IDS perform better in identifying previously unknown attacks [14]. However, anomaly-based approaches suffer from low accuracy and high false positive rates, which limit their reliability.

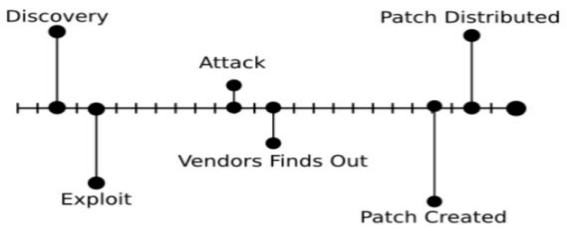

Figure 2 Lifecycle of ZDA

ZDA lacks preset indicators, and prior information is challenging to identify and impossible to predict [15]. The complexities of such attacks have not been fully addressed by traditional machine learning (ML) models and some deep learning (DL) methods [16–18]. Since many models fail to function consistently on new data, insufficient datasets for testing and training are particularly problematic [19]. ZDA detection leverages recent advancements in DL frameworks, including convolutional neural network (CNNs), long short-term memory auto-encoder (LSTM-AE), and generative adversarial networks (GANs). However, these approaches face challenges related to generalizability and scalability [20–22]. In this context, federated learning (FL) is emerging as a promising approach to enhance the anomaly detection performance of ZDAs across various scenarios in the internet of things (IoT) and 5G networks [23, 24]. Additionally, another innovative approach involves using GANs to generate synthetic ZDA samples, which can improve classifier training and enhance robustness [25].

Previous research on zero-day malware detection has addressed several significant challenges. Existing IDS, which primarily rely on signature-based or heuristic algorithms, often fail to identify new or sophisticated attacks [26]. ML-based approaches frequently suffer from limited dataset diversity and generalization issues, as they depend heavily on labeled data that may not accurately represent real-world threats, as discussed by [27, 28]. Additionally, adversarial attacks exacerbate these challenges by exploiting vulnerabilities in model training. Moreover, many models designed to handle the complexities of ZDAs lack effective feature selection and processing mechanisms [29, 30].

Traditional detection approaches face significant challenges due to the increasing complexity and frequency of zero-day malware attacks, as well as their ability to exploit unknown vulnerabilities. Existing models often struggle with limited dataset diversity, poor generalization, and the difficulty of accurately replicating real-world zero-day threat scenarios. This study is driven by the urgent need to develop a robust system capable of overcoming these limitations through adversarial learning and enhanced dataset complexity. As a result, the detection of previously unseen malware is improved. The ultimate goal is to create a scalable and adaptable system to combat emerging cyber threats effectively.

This study employed a ResNet-50-based CNN to address these challenges by developing a DL-based IDS for real-time ZDA detection. The proposed approach minimized FPs while enhancing detection accuracy and scalability, enabling it to efficiently manage high network traffic. The primary objective was to improve the prediction and detection of zero-day vulnerabilities using DL methodologies, thereby strengthening IDS in handling emerging cyber threats.

The main aim is to develop and evaluate the deep convolutional zero-day adversarial safety network (DC-nZDASN) framework for generating adversarial malware samples that mimic real-world threats, thereby enhancing zero-day malware detection. This approach focuses on increasing dataset diversity through feature modification, stabilizing training

using deep auto-encoders (AEs), and training the discriminator to recognize malware-specific patterns. This research presents a comprehensive framework for ZDA prediction and detection, incorporating advanced feature selection, model training, and classification techniques. The hybrid approach of random forest (RF) and logistic regression (LR) ensures optimal feature selection, while dataset preparation utilizes a standard scaler and decision tree regression (DTR) for pre-processing. ResNet-50 and long short-term memory (LSTM) are integrated into the training phase to improve prediction performance, with further enhancements achieved using Adam optimization (AO). Additionally, stacking ensemble classification (SEC), LR, gaussian naïve bayes (GNB), decision tree (DT), and support vector machine (SVM) were used for classification.

The paper is organized as follows: Section 2 presents an overview of related works. Section 3 introduces the proposed model, describing its architecture and methodology. Section 4 details the experimental results, providing analysis and interpretation. Finally, Section 5 concludes the paper by summarizing the key contributions and outlining directions for future research.

2.Literature review

This section discussed the related work to provide a more expansive idea focusing on ZDA prediction. Idhammad et al. [31] provided a supervised CNN architecture for edge systems that addressed latency and limited resources. Their technique complemented the studies on dispersed edge network protection, matched research on feature extraction, and underlined the robustness of CNN in distributed denial-of-service (DDoS) traffic detection.

Older models relied heavily on rule-based or statistical techniques that struggled with flexibility and accuracy. IDs have evolved significantly. Lin et al. [32] proposed cluster center and nearest neighbor (CANN), a strategy that combined cluster centers with nearest neighbors to increase detection accuracy and efficiency. It used cluster-based preprocessing for data dimensionality reduction and computation efficiency improvement. The closest neighbor technique ensured precise anomaly classification. Their hybrid approach addressed its limits in standalone clustering or neighbor-based systems by providing a fair trade-off between accuracy and processing efficiency. Their approach was consistent with previous research, emphasizing the need to combine unsupervised and supervised approaches for robust anomaly detection. It influenced subsequent studies focusing on hybrid IDS models for effectively managing large-scale and dynamic network environments.

Staudemeyer [33] explored the application of LSTM-recurrent neural network (RNN) for IDS. Their study demonstrated that LSTM's ability to effectively understand sequential network traffic data enabled the identification of detailed infiltration patterns. By leveraging LSTM’s capability to retain long-term contextual information and capture temporal correlations, the model outperformed traditional IDS strategies in anomaly detection. The research also highlighted the significance of hyperparameter tuning and architectural design in enhancing detection accuracy. This work has contributed to the adoption of advanced RNN architectures in cybersecurity applications, particularly for real-time intrusion detection.

Sarhan et al. [34] studied zero-shot ML techniques in detecting ZDA. It focused on the function of ZDA detection by the significantly low amount of labeled training data for such threats. The proposed methods were able to generalize and identify anomalous behavior by using feature embeddings and transfer learning towards anomalous behavior identification indicative of ZDA. The research showed that zero-shot learning (ZSL) had been beneficial in situations that typical supervised models could not handle. This inability was due to the lack of knowledge that ZSL helped improve cybersecurity resilience.

Jose and Jose [35] developed the convolutional neural network with long short-term memory (CNN-LSTM) IDS, an anomaly- and signature-based IDS for IoT, utilizing both CNN and LSTM. Their work addressed the unique security challenges of IoT systems, where sophisticated detection techniques are necessary due to the vast data volume and diverse attack vectors. By integrating CNN for feature extraction and LSTM for sequence learning, the proposed model effectively detected both known and novel attacks. It demonstrated a promising approach for protecting IoT networks against emerging threats, achieving high detection accuracy and low false-positive rates.

In the context of cybercrime, Alazab et al. [36] explored the concept of crime toolkits and highlighted their role in facilitating online malicious activities. They examined the commercialization of cybercrime and the accessibility of widely available

tools that enable attackers to conduct operations without extensive technical expertise. Their study provided a detailed analysis of various cybercrime toolkits, including their components, functionalities, ease of access, and usage by attackers. This research highlighted the need for robust security measures to combat the threats posed by these toolkits. Their work contributed to a deeper understanding of cybercrime dynamics and the increasing complexity of cyberattack techniques.

Pascanu et al. [37] explored the use of RNNs for malware classification, emphasizing their ability to process sequential data for detecting malicious software. Their research demonstrated RNNs' effectiveness in capturing temporal patterns within malware execution behavior, enabling the identification of complex and previously unknown malware variants. The authors highlighted that RNNs significantly improved detection accuracy compared to traditional signature-based techniques. This study underscored the potential of DL technologies in strengthening cybersecurity, particularly in detecting malware employing evasive tactics, thereby enhancing dynamic and automated malware detection systems.

Arun et al. [38] investigated the detection and simulation of ZDA using advanced DL methods. Their study emphasized the challenges of detecting zero-day vulnerabilities due to the lack of historical data and predefined signatures. To address this, the authors proposed a DL framework based on anomaly detection, integrating CNNs and LSTM networks to analyze network traffic and system logs for patterns indicative of ZDA. Experimental results demonstrated that the proposed framework achieved superior detection accuracy, robustness, and potential real-time application in IDS.

Oluwadare and ElSayed [39] examined unsupervised learning algorithms for detecting ZDA in IDS. Their research focused on identifying previously unseen attacks in the absence of labeled data and highlighted the potential of unsupervised learning approaches to address this challenge. The authors compared various unsupervised techniques, including DL, anomaly detection, and clustering, in ZDA detection. Their findings demonstrated how different strategies handled the evolving nature of cyber threats, outlining their respective strengths and weaknesses. The study concluded that unsupervised learning holds significant promise in enhancing IDS capabilities by identifying new attack patterns without requiring large labeled datasets.

Aljawarneh [40] analyzed technological advancements, security concerns, and emerging challenges in online banking systems. The study identified vulnerabilities threatening the integrity and trustworthiness of online transactions, including phishing attacks, malware, insider threats, and denial-of-service (DoS) attacks. They discussed the growing complexity of cyber threats and the need for robust security solutions to mitigate these risks. The research examined emerging technologies such as anomaly detection systems, encryption techniques, and multi-factor authentication, emphasizing the necessity for continuous innovation and security enhancements to protect online banking systems from evolving digital threats.

Demirel and Sandikkaya [41] introduced a web-based anomaly detection framework using CNNs for ZSL. Their study highlighted the limitations of traditional anomaly detection methods, which rely heavily on large labeled datasets for optimal performance. The proposed framework overcame this constraint by enabling the detection of previously unseen attack classes or anomalies without requiring labeled data. By integrating CNNs, the model improved feature extraction representation and enhanced accuracy in anomaly detection within web environments. This approach proved particularly valuable in dynamic and evolving cybersecurity landscapes, as it achieved scalable and adaptive anomaly detection with minimal prior knowledge of attack patterns.

Bai et al. [42] showed that temporal convolutional networks (TCNs) were more effective in sequential data processing flow than recurrent architectures such as LSTM and gated recurrent units (GRUs). Among the various potential inductive biases offered, TCNs achieved parallelization, flexible receptive fields, and stable gradients that remediated some issues, such as vanishing gradients and sequential computation bottlenecks in pure recurrent models. It has been shown that TCNs achieve higher accuracy and efficiency than conventional recurrent models in tasks such as sequence prediction and classification. This approach utilizes dilated convolutions to achieve large receptive fields, which are essential for real-world time series and sequence modeling, albeit at the cost of computational efficiency.

Roy et al. [43] employed an artificial neural network (ANN) approach to develop an IDS that enhanced

network security. Specifically, the model overcame the limitations of traditional IDS in detecting unknown attacks. The DL capabilities of the ANN model were leveraged to analyze complex patterns in network traffic and distinguish between normal and malicious activities. They demonstrated that ANN is scalable, adaptable to large datasets, and capable of achieving high accuracy in real-world intrusion detection. The study also highlighted that DL-based approaches provide higher detection rates for both known and unknown intrusions compared to conventional methods.

Habibi et al. [44] evaluated multiple CNN architectures and pre-trained models in terms of their performance in malware classification. This study compared the effectiveness of CNN-based architectures against pre-trained malware classifiers that utilized transfer learning for accurate malware classification. The researchers found that pre-trained models leveraged prior knowledge, improving classification performance, particularly on small datasets. Evaluation metrics such as accuracy, precision, and computational efficiency were used to assess model performance. Their experiments demonstrated that pre-trained models generally outperformed standard CNNs in terms of accuracy and robustness, providing valuable insights for enhancing malware detection systems in cybersecurity.

In network security, Hindy et al. [45] explored DL-based methods for efficiently detecting ZDA. They introduced advanced neural network (NN) capabilities that enabled the identification of previously unknown attacks targeting undisclosed vulnerabilities. Their proposed models analyzed network traffic patterns and behaviors to distinguish between normal and malicious activities. Feature selection and preprocessing played a crucial role in improving model accuracy and reducing FPs. The findings indicated that DL-based techniques, particularly CNN-based and LSTM-based approaches, were effective in detecting and mitigating ZDA in critical systems.

Previous studies have highlighted the increasing complexity of IDS and the evolving nature of cyber threats. The integration of DL techniques, such as CNNs, RNNs, and hybrid models, has significantly improved the detection of both known and emerging security risks, including structured query language (SQL) injection, malware, and ZDAs. Traditional detection methods, particularly signature-based approaches, have demonstrated limitations in identifying new and sophisticated threats. In contrast, the adoption of advanced ML models has provided enhanced adaptability to evolving attack strategies. However, challenges persist in addressing insider threats, ensuring scalability, and maintaining regulatory compliance. These challenges emphasize the ongoing need for innovation in IDS development. This review highlights the necessity of advanced IDS technologies, especially in light of the rapidly expanding cybersecurity threat landscape.

3.Materials and methods

The overall architecture of the proposed methodology is illustrated in *Figure 3*, which depicts the ML approach for classification tasks. The process begins with Stage 1, where missing values are managed, and features are scaled using a DTR. In Stage 2, feature importance is analyzed using LR and RF. Stage 3 involves training with LSTM and ResNet-50 models. In Stage 4, classification is performed using stacking ensemble techniques, incorporating various ML algorithms such as DT, Naïve Bayes (NB), SVM, and LR. This section details the feature selection and preprocessing techniques used to develop an ML model for detecting ZDA and the neural network (NN) training procedures.

The methodology begins with feature extraction, identifying and categorizing relevant properties. These features are then normalized, ensuring a consistent dataset suitable for ML algorithms. Using probabilistic and graph-based approaches reported by Yin et al. [46], multiple ZDA pathways are analyzed, enhancing the robustness of the detection model.

Backpropagation neural networks (BPNN) are utilized to analyze complex attack patterns in cloud systems, identifying potential vulnerabilities and attack pathways, as discussed by Swathy and Padmavathi [47]. Features selected using RF techniques undergo further analysis with LR to assess feature significance and enhance prediction accuracy.

Additionally, the hybrid game theory (HGT) method integrates game theory with ML techniques to improve attack detection. Previous studies by Dhanya et al. [48] and subsequent advancements by Akshaya and Padmavathi [49] in phase 2 of their research explore the application of HGT. This combination of methodologies examines attack tactics and decision-making mechanisms, enabling advanced ZDA detection and enhancing the overall performance of the model.

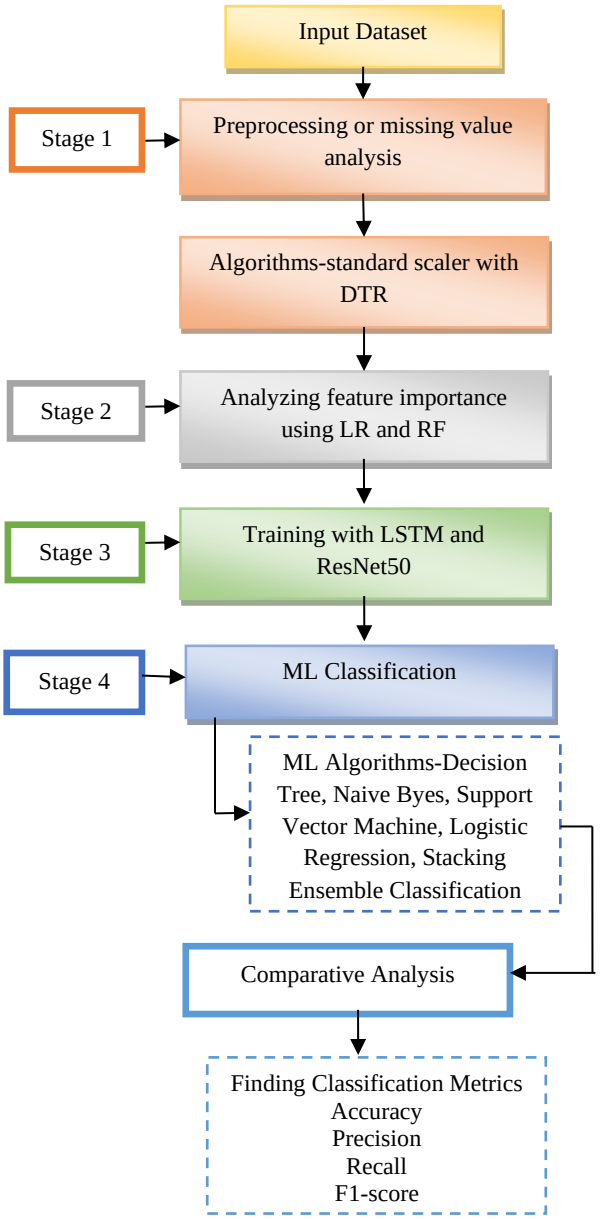

Figure 3 Proposed methodology

3.1Dataset collection

Two datasets have been considered for the experimentation. Dataset 1: national science library-knowledge discovery in databases (NSL-KDD) is obtained from the Kaggle repository and is provided in a comma-separated values (CSV) format, following the suggested methodology for IDS evaluation. NSL-KDD is specifically designed for ZDA detection, aiming to identify new, previously unseen attacks in network traffic [50]. It is an improved version of the KDD Cup 1999 dataset, addressing redundancy and improving the quality of attack samples. This dataset contains labeled network traffic data with 41 features, including protocol type, service, flag, and other statistical measurements. It encompasses multiple attack categories such as DoS, probe, remote-to-local (R2L), and user-to-root (U2R), providing a broad range of feature values for comprehensive IDS evaluation.

The second dataset, Celosia, is explicitly designed for cybersecurity research, focusing on detecting zero-day and anomaly-based attacks. It includes various network traffic attributes such as traffic volume, packet headers, flow durations, and payload entropy. Unlike NSL-KDD, the Celosia dataset integrates both labeled and unlabeled data, offering a more realistic simulation of diverse network conditions by covering extensive traffic patterns and attack behaviors [51]. This makes it particularly useful for anomaly-based IDS studies. The concept of ZDA refers to vulnerabilities exploited before any security mechanisms are in place, making them one of the most challenging issues in cybersecurity. To enhance predictive accuracy in anomaly-based IDS, Celosia has been utilized in research studies such as those conducted by Tavallaee et al. [52].

This research introduces a novel method called DC-nZDASN, which generates synthetic malware samples to distinguish them from real malware. The data generated through random sampling closely resembles the original dataset but retains distinct characteristics. Unlike genuine data, these synthetic samples have altered attributes, making them useful for improving detection models. The proposed model utilizes both real-world malware characteristics and synthetically modified data created by DC-nZDASN. This approach stabilizes the training process and enhances feature extraction. By learning malware traits, the model generates generalized data and continuously refines the training dataset before the actual training phase. As illustrated in *Figure 4*, the trained discriminator within the DC-nZDASN framework effectively identifies malware characteristics, improving the robustness of the detection system.

DC-nZDASN improves malware detection, particularly for ZDA, by integrating features and preprocessing techniques to enhance its model performance. Key features include behavioral indicators and signature-like traits. In preprocessing, the standard scaler, along with DTR, ensures the optimal data preparation. Feature selection uses RF with LR to refine the input by highlighting the most

relevant features. The model leverages residual network (ResNet50), LSTM, and CNN to enhance prediction accuracy. Additionally, it employs classification algorithms such as DT, SVM, LR, and GNB to further improve performance. These techniques collectively contribute to increased accuracy.

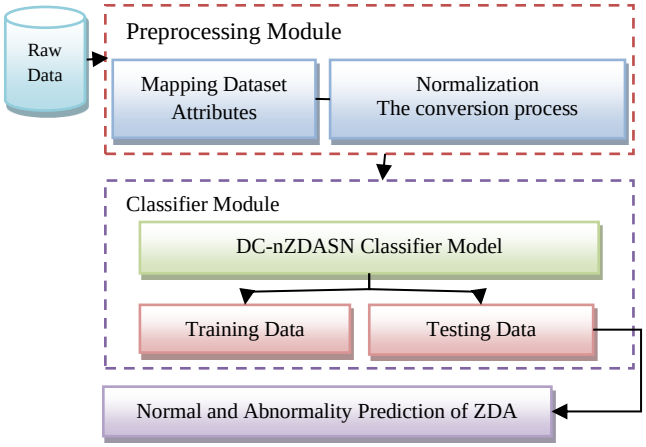

Figure 4 Flow diagram of DC-nZDASN Model

Moreover, the model introduces novel characteristics by generating synthetic malware occurrences, which strengthens its capability to identify emerging threats more effectively.Several solutions are proposed to address DC-nZDASN's class imbalance. Under-sampling the majority class helps balance the dataset using data-level strategies such as the synthetic minority over-sampling technique (SMOTE), which oversamples the low class. Algorithm-level adjustments include setting class weights to find the misclassification in minority classes and incorporate cost-sensitive learning. Evaluation metrics like precision, recall, F1-score, and area under the curve - receiver operating characteristic (AUC-ROC) ensure the model performance. This approach mitigates the impact of class imbalance. Additionally, RF help improve prediction accuracy by aggregating results from multiple balanced subsets of the data, ensuring a more representative and unbiased classification.

Furthermore, this method improves the detection of minority classes. These strategies collectively improve the model proficiency in finding ZDA. Despite imbalanced data distributions, it also ensures robustness and reliability. *Figure 5* depicts the diagrammatical representation of ZDA prediction architecture.

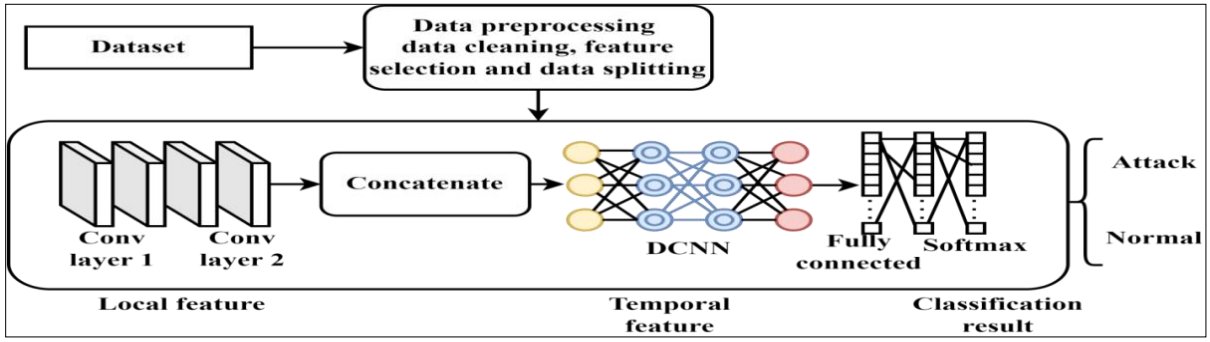

Figure 5 ZDA prediction architecture

3.2Data preprocessing

Data preparation is a crucial step in the data analysis process. Missing values and noise are common challenges in data analytics, often reducing the quality and reliability of the dataset. Data preprocessing plays a vital role in improving the efficiency and accuracy of data mining results by handling these impurities. Furthermore, applying ML techniques to a well-preprocessed dataset is essential for ensuring reliable results and accurate predictions. Proper preprocessing enhances the model's ability to extract meaningful patterns and improve overall performance.

In DC-nZDASN, several enhancements have been made to standard algorithms to improve performance, reduce complexity, and address ZDA detection challenges. The model integrates architectures like ResNet50 and LSTM to capture complex patterns and dependencies in malware data. Additionally, data preprocessing techniques, including standard scaling with DTR and synthetic data generation, enhance data quality and introduce variability crucial for generalization. These modifications collectively improve the model’s ability to accurately detect ZDAs, making DC-nZDASN a reliable solution for cybersecurity applications.

However, misclassifications typically occur in high-noise inputs or attack patterns closely resembling benign behavior. Key challenges include limited training data representations, feature overlap between benign and malicious samples, and adversarial attacks exploiting model weaknesses. Additionally, the evolving nature of cyber threats complicates detection. Addressing these issues requires diverse training datasets, improved noise-handling, refined feature extraction, and continuous learning to adapt to emerging attack patterns.

3.2.1 Label encoding method

Label encoding aims to make the labels legible by the machines that convert them into a numerical representation. ML algorithms are capable of utilizing these labels appropriately. An essential part of this algorithm is the preparation of organized datasets for supervised learning.

3.2.2 Standard scaler method

Standard Scaler represents a data preprocessing technique that standardizes features by removing the mean and scaling features to unit variance. Standard Scaler normalizes data by transforming it to have a mean of 0 and a standard deviation of 1, making it suitable for ML algorithms that are sensitive to feature magnitude, such as SVM and LR. By standardizing features, Standard Scaler ensures that all variables contribute equally to the model's performance, especially when they have different ranges or units, preventing any single feature from dominating the learning process.

The goal is to remove the outliers and scale the features to set the uniform variance. Equation 1 shows the calculation of standard score (Ds).

$$Ds = z\frac{x-\mu}{\sigma} \quad (1)$$

x: The original value of the feature to be standardized.
μ: The feature values in the training set are averaged to get this mean.
σ: It represents the dispersion in the training set's feature value.
z: This is the feature's standardized value or standard score.

The purpose of Equation 1 is to get consistent with different features and performance quality with the algorithms that depend on the scaled data. Many ML estimators depend on the standardized dataset. If the features differ from a normal distribution, the performance is poor. In a data-partitioning setup, the test set is held distinct from the training set, in which the algorithm is accurate. Values in training data, underlying logic, and algorithm features are used to generate the training model. Uniformity across dimensions is the goal of normalization.

Equation 2 shows the attributed information gain.

$$Gain(A) = Info(D) - Info\ A(D) \quad (2)$$

$Gain(A)$: This represents the information gain of attribute *A*. It measures the number of attributes *A* and reduces the dataset *D* entropy (uncertainty).
$Info(D)$*:* This is the entropy of dataset *D* before splitting on attribute *A*.
$Info\ A(D)$: This is the weighted entropy of dataset *D* after splitting on attribute *A*.
Entropy reduction due to dataset splitting to D on attribute A is measured by formula (2). A higher value of this indicates attribute A reduces uncertainty better.

Equation 3 presents the preprocessing information entropy.

$$Info(D) = Entropy(D) = -\sum_j p(j|D)logp(j/d) \quad (3)$$

$Info(D)$ or $Entropy(D)$: This refers to the entropy of dataset *D*. Entropy measures the amount of uncertainty or impurity in the dataset.
$p\ (j|D)$*:* This is the likelihood of class j in dataset *D*.
$logp(j/d)$*:* This is the logarithm of the probability of class *j* in dataset *d*. The logarithm is usually taken in base 2 for the entropy calculations.
Equation 3 calculates the entropy of the dataset D, which quantifies the impurity or uncertainty in the data.

Distribution information entropy is shown in Equation 4.

$$InfoA(D) = \sum_{i=1}^{v}\frac{n_i}{n}Info(Di) \quad (4)$$

$InfoA(D)$: This is the entropy after splitting dataset D on attribute A.
v: This represents the number of distinct values or the partitions created by attribute A.
n_i: This is the number of occurrences in partition Di.
n: This is the total number of occurrences in dataset D.
$\frac{n_i}{n}$: This is the weight of partition Diin related to the total dataset D.

$Info(Di)$: This is the entropy of partition Di.
The entropies of the subsets (Di) formed by the split of the dataset D over the attribute A are weighted averaged. Formulas (2), (3) and (4) are essential

when creating DTs as they are used to find and optimize the splits in the data to give you the highest predictive power.

3.3Feature importance

Step two involves training the network to use the feature importance strategy. The feature importance technique establishes the connection between the attribute and the target set. The dataset is where the input dataset is subjected to the RF technique. Using the LR approach, predicted dataset attributes appear often. Once the dataset has been preprocessed, the hybrid approach combines RF and LR algorithms to choose the features.

Prioritizing the most significant traits during training is achieved with RF, which ranks the features by their value. LR improves the model's accuracy and efficiency by identifying the most predictive characteristics.

3.3.1 RF

Feature space *X* of *M* dimensions hold the sample dataset *D*. After randomly selecting many high-quality trees *P* from a forest, the number of good and uncorrelated trees (*Q*) is determined. The five steps below outline the building of improved RF using *X* and *Q* uncorrelated high-performance trees.

1. *K* in-of-bag (IOB) data subsets are denoted as IOB1, IOB2, and IOBK by unsystematically sampling *D* with substitution and the bagging technique.
2. Assign the evaluation value to each IOB data subset IOB_i and use it to create a tree classifier. This procedure is repeated after harvesting and treating each tree.
3. Arrange these *K* trees by area under the curve (AUC)
4. Choose the best *P* trees that perform well based on their AUC scores.
5. RF construction is improved to determine whether these P trees' estimated probabilities are correlated.

3.3.2 LR

LR is a statistical method used to analyze and quantify the relationship between a dependent variable and one or more independent variables. LR estimates the probability of a particular outcome using the logistic function, making it widely applicable in classification tasks such as binary and multi-class prediction. LR model is described in Equation 5.

$$P_i = \frac{1}{1+exp(-\beta 0-\sum_{j=1}^{k}\beta_i x_{ij})} \quad (5)$$

P_i : This is the predicted probability of the dependent variable being 1 (success) for the observation *i.*

$\beta 0$: This is the intercept term of the model.

β_i: This is the coefficients for independent variables x_{ij}. Each β_i measures the impact of j^{th} independent variable on the log odds of the dependent variable.

x_{ij}: This is the independent variables for x i^{th} observation.

exp: This is the exponential function, often denoted as e

When applying the logit transformation to Equation 5, the linear relationship between logit (*pi*) and explanatory variables is presented in Equation 6.

$$logit(p_i) = log\ (\frac{p_i}{1-p_i}) \quad (6)$$

$logit(p_i)$: This is the logit function of p_i, which transforms the probability p_i Into log odds.

Log : This is the natural logarithm (base).

3.4Training with LSTM and ResNet50

3.4.1 Bidirectional long short-term memory (Bi-LSTM) and LSTM

RNNs primarily rely on short-term memory, enabling them to process sequential data by retaining information from previous time steps and applying it to the current state. However, standard RNNs struggle with retaining long-term dependencies due to the vanishing gradient problem, where gradients diminish over multiple layers, leading to ineffective learning in deeper layers with low weights. This issue often results in the loss of essential information during training.

Compared to traditional feedforward RNNs, LSTM networks are more effective in preserving long-term dependencies. LSTM employs a forget gate mechanism, allowing it to selectively retain or discard information, which enhances its performance over standard RNNs. Due to this gate-controlled memory management, LSTM outperforms conventional RNNs in handling sequential data.

Equations 7 to 11 define the mathematical operations used to compute the hidden states (b_q) of an LSTM unit, based on the values of its input, forget, and output gates.

$$f_q = \sigma\ (\omega_f.[b_{q-1}, x_q] + a_f) \quad (7)$$

f_q: At the time, disregard the gate activation vector in step q.

σ: A sigmoid activation function produces integers from 0 to 1.

ω_f: Weighted matrix for the forget gate.

b_{q-1}: Past hidden state.
x_q: Current input at the time step.
a_f: Bias for the forget gate.

$$i_q = \sigma(\omega_i.[b_{q-1}, x_q] + a_i) \quad (8)$$

i_q: Input gate activation vector at time step q.
ω_i: Weights for the input gate.
a_i: Bias for the input gate.

$$o_q = \sigma(\omega_o.[b_{q-1}, x_q] +) \quad (9)$$

o_q: Output gate activation vector at time step q.
ω_o: Weights for the output gate.
a_o: Bias for the output gate.

$$\tilde{c}_q = tanh(\omega_c.[b_{q-1}, x_q] + a_C) \quad (10)$$

$\tilde{c}_q$: The state vector of candidate cells at time step q
tanh: Activation function that uses hyperbolic tangents and returns values between -1 and 1
ω_c: Considerations for the potential cell state
a_C: The potential bias of cell state

$$c_q = f_q.c_{q-1} + i_q.\tilde{c}_q \quad (11)$$

c_q: Cell state vector at time step q.
c_{q-1}: Previous cell state vector.
.: Element-wise multiplication.

$$b_q = tanh(c_q) \quad (12)$$

b_q: Hidden state vector at time step q.

This recommended technique includes the usage of LSTM. The Foundation of LSTM is RNN, which looks at the sequence in both directions. Load forecasting, categorization, computer vision, and energy consumption prediction (ECP) are the areas in which LSTM has excelled.

3.4.2 CNN with LSTM

Utilizing the combined CNN and Bi-LSTM architecture proves highly effective in predicting ZDA under cyber security. Preprocessing data extracts the relevant features and label representation. LSTM captures temporal reliance, and CNN extracts spatial patterns from input data, such as network traffic logs. These networks are fused to integrate the spatial and temporary features that facilitate the model's ability to detect anomalies or deviations from the expected indication of ZDA. This model demonstrates robustness and accuracy in identifying previously unseen cyber threats through training, validation, and continuous monitoring. This model has been a valuable tool for proactive cyber security measures.

3.4.3 Applying with ResNet

ResNet-50, a residual neural network (ResNet) variant, is a 50-layer deep architecture known for its ability to improve accuracy by filtering more data, as discussed by Shaikh and Gupta [53]. ResNet models commonly incorporate skip connections, enabling the network to bypass one or more layers, effectively mitigating the vanishing gradient problem. These models frequently include double or triple-layer skipping, integrating nonlinear activations rectified linear unit (ReLU) and batch normalization to enhance stability and training efficiency.

Additionally, the Highway Network introduces an extra weight matrix to dynamically adjust the contribution of different layers, further improving DL performance. ResNet-50 follows a convolutional block sequence architecture, ending with an average pooling layer to refine feature extraction.

To evaluate ResNet's predictive accuracy, Mean Squared Error (MSE) is used as a performance metric. The ResNet model is applied to estimate traffic volumes across different road segments, aiming to achieve the closest alignment with real-world data. Consequently, the training objective of ResNet-50 minimizes MSE, as formulated in Equation 13.

$$MSE = \frac{1}{n}\sum_{i=1}^{N}(y_i)^2 \quad (13)$$

n: Total number of samples.
N: In the context of the given equation, Nis is the total number of samples, typically n. This represents the mistake or misrepresentation in the original data.
y_i: Actual value for i^{th}sample.
$(y_i)^2$: Squared difference between the actual and predicted values for i^{th} sample.

3.4.4 Bi-LSTM with ResNet

Here, training models provide the strategy for predicting domestic energy uses. These are as under:
In data preprocessing, mean value *m* is substituted for missing values in the column for the dataset's significant number of missing values. The provided formula is used to normalize the whole values in a min-max scaler with a range of [0, 1] to deal with the adhered data as per Equation 14.

$$m_n = \left[\frac{M - M_{min}}{M_{max} - M_{min}}\right] \quad (14)$$

m_n: Normalized value at a given time.
M: Original value at a given time step.
M_{min}: Minimum value.

M_{max}: Maximum value.

Normalized dataset values are M_{max}, M_{min} and m_n, where M is given the time step value. Preparing data helps to reduce time spent on computations. This ensures that no critical information is lost.
Before feature selection, the dataset is partitioned into features and labels using the sliding window technique. A labeled column with its current value utilizes the historical values as features. A 60-value window is required for the proposed method.

A model's network efficiency is improved or decreased depending on the design choices made during construction. Modifications to the kernel's size filter count and several methods are used throughout the test to validate the effect. This parameter has an impact on data-dependent network performance.

CNN uses the ReLU activation function in its last layer after the time-distributed convolution at each layer's beginning. After the CNN layer, the output is sent via the dense, LSTM, and time-distributed flattening layer. CNN-LSTM model layer has 64 convolution filters, 75 LSTM units, and one dense unit. With max pooling, there is precisely one kernel per layer. *Figure 6* shows the proposed method for determining the energy use.
1) In the first stage, information is cleaned and organized.
2) Features and labels are retrieved and analyzed before the data is separated into training and testing sets.
3) Each model's network architecture is checked using the various performance indicators.
4) Future energy consumption is accurately predicted by selecting the optimal model and architecture.

Hyperparameter tuning process for DC-nZDASN involves systematically optimizing the key parameters such as learning rate, batch size, number of layers and neurons (CNN layers (2, 4, 6, 8), LSTM units: (50, 100, 150, 200), dropout rate (0.1, 0.2, 0.3, 0.4, 0.5), activation functions (ReLU, Sigmoid, Tanh, Leaky ReLU), optimizers and number of epochs (10, 20, 50, 100, 200). This process uses a grid search (GS) for the initial comprehensive evaluation and random search (RS) for the efficiency and Bayesian optimization for fine-tuning. Criteria for selecting the optimal parameters include validation accuracy loss; F1-Score and ROC-AUC are used to prevent the overfit. Combining these techniques makes the model robust and performs high in achieving superior detection rates for ZDA.

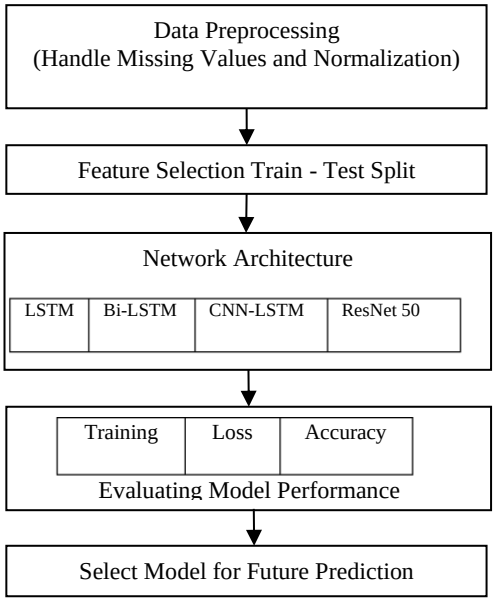

Figure 6 Proposed hybrid model for training and testing

Hyperparameter tuning involves techniques like GS, RS, Bayesian optimization, and K-Fold Cross-Validation. GS systematically evaluates the combinations of parameters (e.g., learning rates (1e-5, 1e-4) with batch sizes (32, 64) but are computationally expensive. RS randomly samples parameter combinations, offering rapid evaluations. Bayesian optimization uses probabilistic models such as Gaussian Processes to balance exploration and exploitation of efficient identification of optimal hyperparameters. K-Fold Cross-Validation splits data into k subsets, trains on k-1, and validates on one, ensuring robust generalization. Data is divided into training (60-70%), validation (10-20%), and testing (10-20%) sets, refining model performance and ensuring accurate evaluation.

3.5ML classifications

3.5.1 Decision tree algorithm

DT shows the attribute test as an internal node, test results as a branch and classes as a leaf node. The root node is shorthand for the tree's first node. Information gain, gain ratio, and G_iN_i The index is the most prevalent attribute selection metric used in the DT method. S is assumed as data collection, unique value m for the class label features, and g distinct classes are indicated by p_i (i, 1, 2 ... g). p_i is equal to the number of items in I class. Equation 15 gives the estimated data for each given sample for classification.

$$I(se_1, se_2, se_3, \ldots se_g) = \sum_{i=1}^{g} p_i log_2(p_i) \quad (15)$$

se_1: Number of samples in i[th] class.
se: Total number of samples.
p_i: Probability of a sample belonging to i[th] class, calculated as $p_i = \frac{se_1}{se}$
Entropy: A measure of impurity or disorder. Lower entropy means a real subset.
Where pi = se_i/ se is the chance of any sample belonging to the population.

ADT sorts a test tuple into many possible categories derived from the set of training tuples. A tuple is an item of information whose properties have been defined. D is the dataset that uses DT representation and contains T sets of training tuples as per Equation 16.

$$D => Q = [q_1, q_1, \dots, q_n] \quad (16)$$

D: The entire dataset.
Q: A set of tuples$[q_1, q_1, \dots, q_n]$.
q_i: Each tuple in the dataset, where i ranges from 1 to n (the total number of tuples).

The maximum number of tuples in the dataset is denoted by n.

3.5.2 Classification using linear SVM

A support vector (SV) is created to better categorize the new data sets by amplifying the information in the training set S. SVMs rely on recognizing the patterns. On data classification, SVM seeks the precise hyperplane that divides the overall data points into their respective classes. One of the two categories is hyperplane with motion.

Hence, the likelihood of incorrect classification is reduced; SVM in data is used to train or test the set. The SVM method is highly recommended because of its excellence in overall efficiency. Binary classification is the possible use of SVM. A proper multiclass approach is required to deal with the multiple data classification and identification classes. A simple binary classification issue is presented in Equation 17.

$$T = \{(X_1, Y_1), (x_2, y_2), (x_3, y_3) \dots \dots (x_n, y_n)\}, y_i \epsilon \{-1,1\}, x_i \epsilon R^d, \quad (17)$$

x_i : This is the data point.
y_i: Represents the corresponding label.
n: Denotes the number of training samples.
d:Indicates the dimensionality of data points.
$X_i$is the data point, and y_i is the corresponding label.
Training data set member is denoted as *n*. Linear SVM determines the ideal separation margin by assessing the optimization goal, as presented in Equations 18 and 19.

$$min\{\frac{1}{2}|a|^2 + p\sum_{i=1}^{l} s_i \geq 0 \quad (18)$$

$$subject\ to y_i(a^T w_i + b) \geq 1 - s_i, i = 1,2,3,4 \dots \dots l \quad (19)$$

a:Is the normal vector to the hyperplane.
b: Is the bias term (offset)
s_i: The slack variables allow some misclassification (for non-separable cases)
Parameter p regulates the trade-off between margin maximization and classification error minimization.

Normal vectors a, seal amount b, and S_i Are the positive slack variables using a Lagrangian multiplier?a_i. Minimum issues are up to the level that provides an optimum solution, as referred by Karush-Kuhn-Tucker criteria. If a_i is more significant than zero, information associated w_i is called SV. As a result, optimal values of a and b in the hyperplaneare are used to create the linear discriminative function, as shown in Equation 20.

$$f(x) = sgn(\sum_{i=1}^{l} \alpha_i z_i w_i^T w + b) \quad (20)$$

The liberated dual form of Equation 21 is:

$$max\left(\sum_{i,j=1}^{n} a_i, a_j, y_i, y_j\, x_i^T x_j\right) \quad (21)$$

To solve the issues, the utility of quadratic programming methods and Karush-Kuhn-Tucker is addressed in Haeser and Ramos [54]. Based on the findings, the possibility of denoting A as a linear combination of training vectors and b as the mean of SV is shown in Equations 22 and 23.

$$A = \sum_{i=1}^{l} \alpha_i z_i w_i \quad (22)$$

$$b = \frac{1}{N_{sv}} \sum_{i=1}^{N_{sv}} (A w_i - z_i) \quad (23)$$

N_{sv} Represents the number of supports in vector. While SVM provides an alternative method to recognize the categories, it visually involves considerable optimization and pairwise distance calculation time. When filtering the neighbors for sample issues, the local vector machine maintains the distance function of the neighbor group. In contrast to the closest neighbors and SVM, the efficient strategy undertaken by SVM performs better with multiclass data. This research classifies surface electromyography (SEMG) with a linear SVM classifier.

3.5.3 Naive bayes classifier

To simulate $B\left(\frac{M}{N}\right)$, M is the feature vector, and Y is the label per Equation 24.

$$B\left(\frac{M}{N}\right) = B\left(\frac{M_1}{N}\right) B\left(\frac{M_2}{N}\right) \ldots, B(\frac{M_Y}{N}) \quad (24)$$

Number of parameters is $nk + k - 1$. Data has to be practiced: M is the feature matrix, y_1 and y_n are the labels. Equations 25 and 26 present the above content as follows:

$$classprior: B(n) = |\{i: n_i = N\}|/y \quad (25)$$

$$Likelihood: pm_i(y) = \frac{B(m_i, n)}{B(N)} = \frac{\left|\{i: m_{ij} = m_i, n_i = n\}\right|/y}{|\{i: n_i = n\}|/y} \quad (26)$$

NB Classifier-Continuous: The below area improbability density function graphs between m_1 and m_2 denotes the likelihood that a random variable has a value between $m_1\ and\ m_2$, as shown in Equation 27.

$$f(m) = \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(m-\mu)^2}{2\sigma^2}} \quad (27)$$

μ: Mean of the feature values.
σ: Standard deviation of the feature values.
m: Feature value.
This function represents the likelihood of feature value m, the given meanμ, and standard deviationσ.

3.5.4 Gaussian parameter estimation

Parameters are the Gaussian distribution's mean σ and variation.σ^2. The given observations $m_1, \ldots, m_y$ and the chance of those observations for a given σ^2 assumed from Gaussian distribution is given in Equation 28.

$$B\left(m_1, \ldots, m_y \middle| \mu, \sigma^2\right) = \prod_{y-1}^{y} \frac{1}{\sqrt{2\pi\sigma}} e^{-\frac{(m_n-\mu)^2}{2\sigma^2}} \quad (28)$$

m_n: Observation n.
μ: Mean of Gaussian distribution.
σ^2: Variance of Gaussian distribution.
It represents the product of the probability density function values under each observation in Gaussian distribution, as shown in Equation 29.

$$L(\mu, \sigma) = -\frac{1}{2} N log(\pi\sigma^2) - \sum_{n=1}^{N} \frac{(x_n-\mu)^2}{2\sigma^2} \quad (29)$$

N: Total number of observations.
It combines the logarithm of the Gaussian probability density function for every observation, making it easier to differentiate and find the maximum likelihood estimates (MLE) of μ and $2\sigma^2$.
Taking the derivative of the desired variable and solving determines the values of two, which maximize the log-likelihood. As a result, the MLE of the mean is provided in Equation 30.

$$\mu = \frac{1}{Y} \sum_{y=0}^{y} m_y \quad (30)$$

Equation 31 represents the arithmetic mean of observations. It is derived by setting the derivative of the log-likelihood function μ to zero and solving the function.

MLE of the variance is

$$\sigma^2 = \frac{1}{Y} \sum_{y=0}^{Y} (m_n - \mu^2) \quad (31)$$

This equation represents the variance of observations. It is derived by setting the derivative of the log-likelihood function from σ^2 to zero and solvingσ^2.

3.5.5 LR

LR is a technique for learning functions in which $M = M_1 \ldots M_y$ are the vectors of discrete or continuous variables, and the vectors for discrete-valued is N either f, M, N, or B $\left(\frac{N}{Y}\right)$. This section covers the case when N is a Boolean variable. At last, the investigation is extended to include the case of N taking the small set of possible values. In LR, P $\left(\frac{N}{M}\right)$distribution is assumed to have a parametric shape, and its parameters are derived from the training data. Equation 32 presents the discrete-valued.

$$B(N = 1|m) = \frac{1}{1+exp\ (W_o + \sum_{i=1}^{y} w_i m_i)} \quad (32)$$

By rewriting$\left(\frac{N}{M}\right)$, a convenient linear classification equation is formed. Assigning an NK value which maximizes B $\left(N = \frac{Nk}{M}\right)$ is the standard classification method. On the other hand, N = 0 is determined if the following holds, as per the Equations 33 to 35.

$$1 < \frac{B(N=0|M)}{B(N=1|M)} \quad (33)$$

On combining, these equations become

$$1 < exp\ (w_0, + \sum_{i=1}^{y} w_i m_i) \quad (34)$$

The linear classification rule is obtained by taking the natural log of both sides i as follows: if M satisfies, then N = 0; otherwise, N = 1

$$0 < w_0 + \sum_{i=1}^{y} w_i M_i \quad (35)$$

3.5.6 Stacking ensemble

Data is gathered over extended durations. After taking T real-value samples $x_1, \ldots, x_T$, output is I divided by the time T$(1 \leq I \leq h)$. Time series forecast compares the expected and actual $xw + i$ values using the historical data from samples $x_1 \ldots x_w$ $(w + h \leq T)$ $(1 \leq I \leq h)$. In this expression, two variables are required: w representing the time frame in the historical window and h representing the time frame in the prediction horizon.

Traditionally, the time sequence is divided into three parts. Different patterns are seen in the time sequence. Remains of time series data illustrate the process of extensive irregular data hiding seasonality and information.

Examining the other breakdown patterns, time effects, and other long-term trends are illuminated. The inherent unpredictability of real-world time series is forecasted notoriously. On top of that, checking of stationary time series is carried out.

Any non-stationary time series needs transformation before using the forecasting model. The number of variables that are compared distinguishes the study of univariate and multivariate time series. Data is strictly chronological and present in univariate time series. At each time point, values of several variables are recorded in multivariate time series. Understanding the connection between these factors is essential. Time series predictions are made using a few different methods. ML techniques for nonlinear modeling prediction have become more significant, and the traditional approaches provide adequate results for linear situations.

3.5.7 Ensemble learning

Enhanced classification and regression performance have increased in popularity, and the models have gained traction recently. These techniques combine several learning models to boost the performance of each model. According to early 1990s ensemble learning research, several relatively weak learning algorithms are combined to create robust algorithms. Ensemble learning makes predictions using several learner modules from a single data set. A single prediction is made when all the expert guesses are added together. This technique typically has two stages. Training data is mined for the set of learners, which are later included in a single prediction model. Several forecasts use the diverse information from varied base models to build the composite model. The improved composite model often outperforms the group of individual models. Ensemble learning is valuable in ML. It is possible to interpret the model by systematically searching for the most promising hypothesis. As the datasets are typically restricted, prediction is improved and performed better on unknown data. This complicates the task of theory selection. Ensemble methods need reasonable approximation of unknowable true hypotheses to address this challenge. The above content is provided in Equation 36.

$$y_{ensemble} = \frac{1}{N}\sum_{i=1}^{N} y_i \quad (36)$$

In ensemble learning, predictions from individual base models y_i are combined using techniques like majority voting or weighted average to arrive at the final prediction$y_{ensemble}$.

Systematic difficulty is the local search often used to reduce the number of incorrect predictions made by ML models. Local optimization impedes these types of global searches. Genuine unknown functions are understood better when the local searches launch from different starting points. Bagging is a method where multiple models are created and compared with the objective results. As a result, a majority vote arrives at the final decision. This procedure is known as bagging. Typically, regression makes an average prediction. There is a single conceptual distinction between boosting and bagging. Instead of assigning equal importance to each model, boosting uses weighted voting to decide the final score. Most regression procedures end with the weighted average. Predictions from the underlying algorithms are used to train the combiner algorithm in making the final predictions. This approach has been used in any ensemble approach. The regression problem in this research is handled by using the stacking technique.

$$y_{bagging} = \frac{1}{N}\sum_{i=1}^{N} y_i \quad (37)$$

Equation 37 shows the replacement use; bagging creates several models, training on separate data subsets, and averages their predictions.

For parallelizing computations and efficient matrix operations during DL model training, systematic resources are used for DC-nZDASN, including the high-performance graphic processing unit (GPUs) such as NVIDIAGeForce RTX or Tesla series. A multi-core central processing unit (CPU)'s complements GPU processing for non-GPU tasks such as data preprocessing. Cloud resources like amazon web services (AWS), google cloud platform (GCP), and Azure are used for scalability. While demanding systematic power, malware featuring GPU support, such as AWS EC2 GCP compute engine, is used for DL tasks. Cloud-based solutions like AWS simple storage service (AWS S3) and GCP cloud storage store large datasets and trained models, whereas technologies like Docker and Kubernetes manage ML workflows efficiently in the cloud. Software tools like Tensor Flow, PyTorch, Pandas, NumPy, and Scikit-learn are implemented for model building, while data preprocessing, evaluation, and cloud management tools aid in resource provision and monitoring. This combination of hardware,

cloud, and software resources optimizes the performance and scalability of DC-nZDN's complex requirements.

For parallelizing computations and enabling efficient matrix operations during DL model training, systematic resources are utilized in DC-nZDASN. This includes high-performance graphics processing units (GPUs) such as NVIDIA GeForce RTX or Tesla series, which accelerate model training. Additionally, multi-core central processing units (CPUs) complement GPU processing by handling non-GPU tasks like data preprocessing. To ensure scalability, cloud resources such as AWS, GCP, and Microsoft Azure are employed. For GPU-intensive DL tasks, cloud-based instances like AWS EC2 and GCP Compute Engine provide the necessary computational power. Large datasets and trained models are stored using AWS Simple Storage Service (AWS S3) and GCP Cloud Storage. For efficient ML workflow management, containerization and orchestration tools like Docker and Kubernetes are used. Additionally, software tools such as TensorFlow, PyTorch, Pandas, NumPy, and Scikit-learn support model building, data preprocessing, evaluation, and cloud resource monitoring. This integrated combination of hardware, cloud computing, and software tools optimizes DC-nZDASN's performance, scalability, and resource efficiency, ensuring smooth execution of complex DL workloads.

4.Results and discussion

The system monitors active nodes in the network, identifying those with a high probability of ZDA occurrence. Nodes flagged as potential threats are isolated to prevent further communication and enhance security. MSE values and threshold-based predictions are adjusted to improve ZDA detection and mitigation.

Table 1 outlines the LSTM and ResNet parameters, including input size, padding, activation functions, and step size, providing a structured evaluation of model performance. Features are categorized as numerical, nominal, or binary, with 41 features from the NSL-KDD dataset considered and validated using the DC-nZDASN model. Among them, 18 key features are identified as the most influential in different classification processes.In label training classifiers, standard sample characteristics are used to detect DoS, Probe, R2L, and U2R attacks. Preprocessed test data is fed into the training classifier, which classifies and identifies samples. Detection results are validated through model verification and test comparisons performed in a Windows operating system (OS) environment.

To improve computational efficiency, parallel computing is implemented. The threshold limit for selecting dataset feature values is predicted, and error values are identified. The dataset is partitioned into training and testing sets based on the four attack types (U2R, R2L, Probe, and DoS) for further processing, as detailed in *Table 2*.

The TP, FP, TN, and FN values are shown in *Figure 7*, representing the confusion matrix. The anticipated class for TP is 28, TN is 17, and FP and FN are 0.

Table 1 LSTM with ResNet parameters

Layers	Type	Size	Padding	Activation function	Step	Classification based on feature inputs
Layer 1	Multi-Layer Convolution	1×1, 3×3, 5×5, 7×7	0 1 2	ReLU	1	11 × 11 × 64
Layer 2	Convolution Layer	3×3		ReLU	1	9 × 9 ×64
Layer 3	Multi-Layer Convolution	1×1, 3×3, 5×5, 7×7	0 1 2	ReLU	1	9 ×9 ×128
Layer 4	Convolution Layer	3×3		ReLU	1	7 × 7 × 128
Layer 5	Multi-Layer Convolution	1×1, 3×3, 5×5, 7×7	0 1 2	ReLU	1	7 ×7 ×256
Layer 6	Pooling Layer	3×3	-	ReLU	2	3 ×3 ×256
Layer 7	Fully Connected Layer	-	-	Sigmoid	-	256
Layer 8	Fully Connected Layer	-	-	Sigmoid	-	64
Layer 9	Fully Connected Layer	-	-	Sigmoid	-	5

Table 2 Attack types and corresponding training / testing set

Attacks	Training set	Testing set
U2R	Ps, Buffer Overflow, Rootkit, Load Module	Perl, Ps, Buffer Overflow, Xterm, Sqlattack
R2L	Warezmaster, Phf, Multi-Hop, internet message access protocol (IMAP), Guess	Ftpwrite, Httptunnel, IMAP, Named, Phf, Multi-hop, Send mail, Snmpgetattack, Wxlock, Snmpguess, Warezmaster, Xsnoop.

Attacks	Training set	Testing set
	Password, Ftpwrite, Spy, Warezclient	
Probe	Satan, Portsweep, Nma, Ipsweep	Msacn, Saint, Satan, Nmap, Portsweep, Ipsweep
DoS	Neptune, Smurf, Back, land, Pod, Teardrop	Udpstrom, Smurf, Worm Process Table, Teardrop, Pod, Neptune, Back, Land, Apache2, Mailbomb

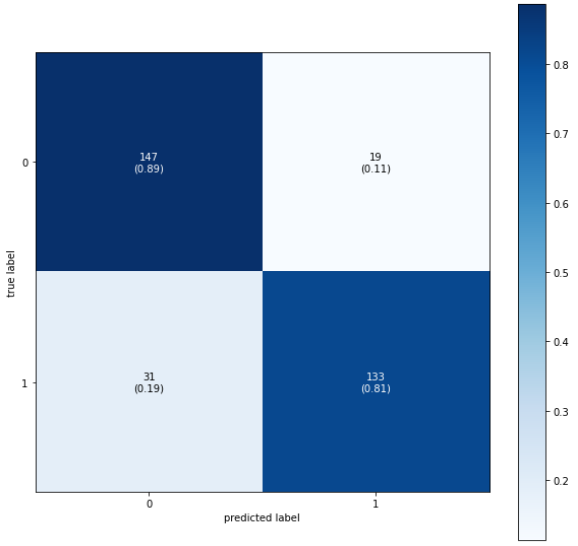

Figure 7 Confusion matrix

4.1Performance metrics

After analyzing the results using a multi-category classification framework, the feature importance accuracy is determined to be 87.58%. Additionally, the one-versus-the-rest method is employed by converting the multi-class classification problem into binary classification sub-problems to enhance prediction accuracy.

The classification components are defined as follows:

a. TP_d: Class D describes both forecast and actuality.
b. TN_d: Some categories of d are the focus of the forecast.
c. FP_d: Category d is the predicted outcome; however, more categories are inside category d in practice.

Every category is used as a positive sample to compute the overall accuracy, precision, and recall. Equation 38 represents the mathematical formulation for accuracy.

$$Accuracy = Numberofsamplescorrectlyclassified\ Numeberofsamplesforallcategories \quad (38)$$

Equation 39 shows that comparing the sample's accuracy with the category's precision is one approach for examining accuracy.

$$Precision_i = TP_dTP_d + FP_d \quad (39)$$

The recall of a specific category measures how well the model correctly identifies instances of category d from the total actual occurrences of that category (Equation 40).

$$Recall_i = \frac{TP_d}{TP_d + FN_d} \quad (40)$$

Equation 41 calculates the F1-score.

$$F1 - score = 2.\frac{Precision.recall}{Nprecison+recall} \quad (41)$$

Findings of training and testing the proposed model on celosia and NSL-KDD datasets are presented in *Table 3*. The experiments use ResNet50 combined with LSTM for the training and testing phases.

Table 3 Training and testing performance of ResNet50-LSTM Model on NSL-KDD and Celosia datasets over 10 epochs

Epoch	Training loss	Validation loss	Training accuracy	Testing accuracy
1	0.0039	0.0626	0.9987	0.9846
2	0.0035	0.0762	0.9989	0.9835
3	0.0030	0.0699	0.9991	0.9853
4	0.0039	0.0643	0.9986	0.9862
5	0.0017	0.0684	0.9995	0.9866
6	0.0030	0.0747	0.9991	0.9858
7	0.0026	0.0887	0.9991	0.9844
8	0.0023	0.0776	0.9992	0.9855
9	0.0023	0.0749	0.9992	0.9868
10	0.0013	0.0767	0.9996	0.9876

The proposed model's training and testing loss parameters are shown in *Figure 8*. The Y-axis displays the loss amount, and the X-axis displays the number of epochs. *Figure 9* shows that CNN-ResNet has trained 10 epochs with testing and training accuracy. As the X-axis displays the epoch numbers, the Y-axis displays the accuracy.

Table 4 presents a comparative analysis of different ML algorithms, including DT, SVM, GNB, LR, and the stacking ensemble classifier, based on their accuracy, precision, recall, and F1-score. The evaluation is conducted on two datasets: NSL-KDD and Celosia, demonstrating the effectiveness of

different classifiers in detecting cyber threats. The stacking ensemble classifier achieves the highest performance across both datasets.

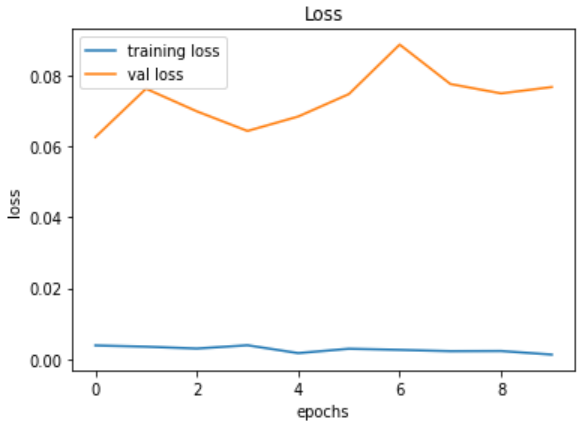

Figure 8 Training and testing loss

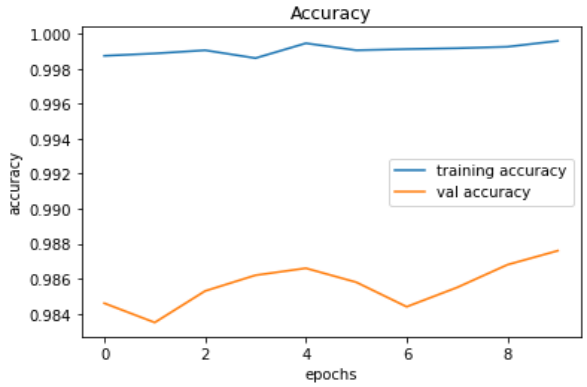

Figure 9 Training and testing accuracy

The proposed model achieves a classification accuracy of 95.09%, surpassing existing techniques in ZDA detection. This high accuracy is crucial for strengthening cybersecurity measures. The key findings of the model are as follows:

1. The model integrates DL architectures such as ResNet50, LSTM, and CNNs, enhancing learning stability and enabling the detection of zero-day threats by recognizing complex internal patterns.
2. The model generates synthetic malware, employs advanced preprocessing techniques, feature selection methods, and ensemble classifiers, significantly improving its ability to detect novel and previously unseen threats.
3. Leveraging cloud-based resources and GPU acceleration, the model ensures efficient data handling, faster training of DL models, and scalability, making it suitable for large-scale datasets and complex security tasks.
4. Strategies such as class weight adjustment, cross-validation, and appropriate evaluation metrics enhance the model's ability to generalize across diverse datasets, reducing overfitting and mitigating bias toward majority classes.
5. The model significantly strengthens threat detection and mitigation strategies, contributing to a more effective and proactive cybersecurity framework.

A complete list of abbreviations is listed in *Appendix I.*

Table 4 Algorithm performance comparison on NSL-KDD and celosia datasets

Dataset	Algorithm	Accuracy (%)	Precision	Recall	F1-score
Dataset 1: NSL-KDD	DT	88	87	90	88
	SVM	83	81	88	84
	GNB	90	91	89	90
	LR	85	83	89	85
	Stacking ensemble classifier	95.9	89.5	88.4	89
Dataset 2: Celosia	DT	91.75	91	92	92
	SVM	71	71	70	71
	GNB	83	87	78	82
	LR	71	72	70	71
	Stacking ensemble classifier	95.9	92.3	91.1	91.7

Figure 10 box plot compares the performance distribution of different classification algorithms based on an evaluation metric such as accuracy. The hybrid model (Stacking ensemble classifier) and SVM exhibit the highest median performance with lower variability, while DT shows the lowest accuracy and higher dispersion, indicating inconsistency in its predictions. The green triangles represent the mean values for each classifier.

Limitations

Despite its effectiveness, the model has certain limitations:

1. The model's performance heavily relies on accurate, diverse, and well-labeled training datasets, which may not always be available.
2. The implementation requires high-performance GPUs and cloud-based infrastructure, making it resource-intensive and potentially costly.
3. The model may struggle to adapt to rapidly evolving attack methods, requiring continuous updates and retraining.
4. Although the model reduces misclassification, completely eliminating false alarms remains a challenge.
5. While the model employs various generalization techniques, emerging threats that differ significantly from training data may still pose detection challenges.

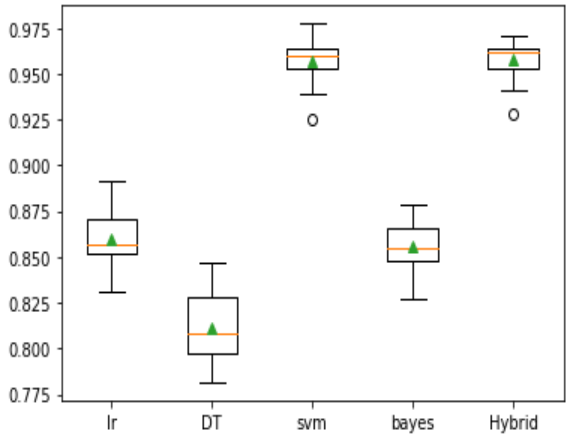

Figure 10 Stacking ensemble classifications (SEC) comparison chart

5.Conclusion and future work

The proposed DC-nZDASN model effectively enhances ZDA detection, achieving a classification accuracy of 95.09%, surpassing traditional methods. By integrating ResNet50, LSTM, and CNN, the model improves learning stability and effectively identifies new and evolving cyber threats. The inclusion of synthetic malware generation, advanced preprocessing techniques, and ensemble classifiers further strengthens its robustness against previously unseen attacks. Utilizing cloud-based resources and GPU acceleration, the model ensures efficient data processing, scalability, and rapid training. Additionally, techniques such as class weight adjustment, cross-validation, and diverse evaluation metrics improve generalization and reduce overfitting. The model achieves a classification accuracy of 95.09%, outperforming other methods and enhancing new threat detection capabilities.

Despite these promising results, continuous research and updates are necessary to address the ever-evolving nature of ZDAs and maintain long-term effectiveness. To improve generalizability, the model requires further testing and validation with diverse datasets and real-time streaming data. Future work will focus on enhancing ZDA classification by integrating DL and ML in IDS, refining adaptability through self-learning mechanisms, and introducing real-time streaming data analysis for dynamic threat detection. Additional improvements include simulation techniques for better training data quality and collaborations with industry experts to ensure real-world applicability.

Acknowledgment

None.

Conflicts of interest

The authors have no conflicts of interest to declare.

Data availability

The dataset used in this study is publicly available and can be accessed at the following links:
Dataset 1: NSL-KDD – https://www.kaggle.com/datasets/hassan06/nslkdd
Dataset 2: Celosia – https://www.kaggle.com/code/mkashifn/celosia-zero-day-attack-detection-demo

Author's contribution statement

Swathy Akshaya: Methodology, data curation, analysis and testing of experimental results, manuscript preparation, plagiarism check, and proofreading. **Padmavathi. G:** Supervised and reviewed the manuscript. All authors have read and approved the final manuscript.

References

[1] Das N, Sarkar T. Survey on host and network based intrusion detection system. International Journal of Advanced Networking and Applications. 2014; 6(2):2266- 9.
[2] Ahmed M, Mahmood AN, Hu J. A survey of network anomaly detection techniques. Journal of Network and Computer Applications. 2016; 60:19-31.
[3] Ahmad R, Alsmadi I, Alhamdani W, Tawalbeh LA. Zero-day attack detection: a systematic literature review. Artificial Intelligence Review. 2023; 56(10):10733-811.
[4] Bou-harb E, Debbabi M, Assi C. Cyber scanning: a comprehensive survey. IEEE Communications Surveys & Tutorials. 2013; 16(3):1496-519.
[5] Soltani M, Ousat B, Siavoshani MJ, Jahangir AH. An adaptable deep learning-based intrusion detection system to zero-day attacks. Journal of Information Security and Applications. 2023; 76:103516.
[6] Khraisat A, Gondal I, Vamplew P, Kamruzzaman J. Survey of intrusion detection systems: techniques,

datasets and challenges. Cybersecurity. 2019; 2(1):1-22.
[7] Kumar GS, Kumar RK, Kumar KP, Sai NR, Brahmaiah M. Deep residual convolutional neural network: an efficient technique for intrusion detection system. Expert Systems with Applications. 2024; 238:121912.
[8] Hubballi N, Suryanarayanan V. False alarm minimization techniques in signature-based intrusion detection systems: a survey. Computer Communications. 2014; 49:1-7.
[9] Ibrahim HB, Aslan HK, Elsayed MS, Jurcut AD, Azer MA. Anomaly detection of zero-day attacks based on CNN and regularization techniques. Electronics. 2023; 12(3):1-18.
[10] Bhuyan MH, Bhattacharyya DK, Kalita JK. Network anomaly detection: methods, systems and tools. IEEE Communications Surveys & Tutorials. 2013; 16(1):303-36.
[11] Verma P, Bharot N, Breslin JG, O'shea D, Vidyarthi A, Gupta D. Zero-day guardian: a dual model enabled federated learning framework for handling zero-day attacks in 5G enabled IIoT. IEEE Transactions on Consumer Electronics. 2023; 70(21):3856-66.
[12] Peppes N, Alexakis T, Adamopoulou E, Demestichas K. The effectiveness of zero-day attacks data samples generated via GANs on deep learning classifiers. Sensors. 2023; 23(2):1-21.
[13] Creech G, Hu J. A semantic approach to host-based intrusion detection systems using contiguousand discontiguous system call patterns. IEEE Transactions on Computers. 2013; 63(4):807-19.
[14] Mukkamala S, Janoski G, Sung A. Intrusion detection using neural networks and support vector machines. In proceedings of the international joint conference on neural networks 2002 (pp. 1702-7). IEEE.
[15] Bajaj K, Arora A. Dimension reduction in intrusion detection features using discriminative machine learning approach. International Journal of Computer Science Issues. 2013; 10(4):324-8.
[16] Lecun Y, Bengio Y, Hinton G. Deep learning. Nature. 2015; 521(7553):436-44.
[17] Farahnakian F, Heikkonen J. A deep auto-encoder based approach for intrusion detection system. In 20th international conference on advanced communication technology 2018 (pp. 178-83). IEEE.
[18] Hnamte V, Nhung-nguyen H, Hussain J, Hwa-kim Y. A novel two-stage deep learning model for network intrusion detection: LSTM-AE. IEEE Access. 2023; 11:37131-48.
[19] Nagasundari S, Honnavali PB. SQL injection attack detection using ResNet. In 10th international conference on computing, communication and networking technologies 2019 (pp. 1-7). IEEE.
[20] Shun J, Malki HA. Network intrusion detection system using neural networks. In fourth international conference on natural computation 2008 (pp. 242-6). IEEE.
[21] Alshehri A, Badr MM, Baza M, Alshahrani H. Deep anomaly detection framework utilizing federated learning for electricity theft zero-day cyberattacks. Sensors. 2024; 24(10):1-19.
[22] Sakthimurugan S, Kumaar S, Vignesh V, Santhi P. Assessment of zero-day vulnerability using machine learning approach. EAI Endorsed Transactions on Internet of Things. 2024; 10:1-6.
[23] Aburomman AA, Reaz MB. A novel SVM-kNN-PSO ensemble method for intrusion detection system. Applied Soft Computing. 2016; 38:360-72.
[24] Alazab A, Khresiat A. New strategy for mitigating of SQL injection attack. International Journal of Computer Applications. 2016; 154(11):1-10.
[25] Ji SY, Jeong BK, Choi S, Jeong DH. A multi-level intrusion detection method for abnormal network behaviors. Journal of Network and Computer Applications. 2016; 62:9-17.
[26] Butun I, Morgera SD, Sankar R. A survey of intrusion detection systems in wireless sensor networks. IEEE Communications Surveys & Tutorials. 2013; 16(1):266-82.
[27] Shiravi A, Shiravi H, Tavallaee M, Ghorbani AA. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. Computers & Security. 2012; 31(3):357-74.
[28] Skaruz J, Seredynski F. Recurrent neural networks towards detection of SQL attacks. In international parallel and distributed processing symposium 2007 (pp. 1-8). IEEE.
[29] Elsherif A. Automatic intrusion detection system using deep recurrent neural network paradigm. Journal of Information Security and Cybercrimes Research. 2018; 1(1):21-31.
[30] Osa E, Orukpe PE, Iruansi U. Design and implementation of a deep neural network approach for intrusion detection systems. e-Prime-Advances in Electrical Engineering, Electronics and Energy. 2024; 7:1-6.
[31] Idhammad M, Afdel K, Belouch M. Semi-supervised machine learning approach for DDoS detection. Applied Intelligence. 2018; 48:3193-208.
[32] Lin WC, Ke SW, Tsai CF. CANN: an intrusion detection system based on combining cluster centers and nearest neighbors. Knowledge-Based Systems. 2015; 78:13-21.
[33] Staudemeyer RC. Applying long short-term memory recurrent neural networks to intrusion detection. South African Computer Journal. 2015; 56(1):136-54.
[34] Sarhan M, Layeghy S, Gallagher M, Portmann M. From zero-shot machine learning to zero-day attack detection. International Journal of Information Security. 2023; 22(4):947-59.
[35] Jose J, Jose DV. AS-CL IDS: anomaly and signature-based CNN-LSTM intrusion detection system for internet of things. International Journal of Advanced Technology and Engineering Exploration. 2023; 10(109):1-18.
[36] Alazab A, Abawajy J, Hobbs M, Layton R, Khraisat A. Crime toolkits: the productisation of cybercrime. In 12th international conference on trust, security and

privacy in computing and communications 2013 (pp. 1626-32). IEEE.
[37] Pascanu R, Stokes JW, Sanossian H, Marinescu M, Thomas A. Malware classification with recurrent networks. In international conference on acoustics, speech and signal processing 2015 (pp. 1916-20). IEEE.
[38] Arun A, Nair AS, Sreedevi AG. Zero day attack detection and simulation through deep learning techniques. In 4th international conference on cloud computing, data science & engineering (confluence) 2024 (pp. 852-7). IEEE
[39] Oluwadare S, Elsayed Z. A survey of unsupervised learning algorithms for zero-day attacks in intrusion detection systems. In the international FLAIRS conference proceedings 2023 (pp. 1-3). FLAIRS.
[40] Aljawarneh SA. Emerging challenges, security issues, and technologies in online banking systems. In online banking security measures and data protection 2017 (pp. 90-112). IGI Global.
[41] Demirel DY, Sandikkaya MT. Web based anomaly detection using zero-shot learning with CNN. IEEE Access. 2023; 11:91511-25.
[42] Bai S, Kolter JZ, Koltun V. Convolutional sequence modeling revisited. ICLR Workshop. 2018 (pp. 1-20).
[43] Roy SS, Mallik A, Gulati R, Obaidat MS, Krishna PV. A deep learning based artificial neural network approach for intrusion detection. In mathematics and computing: third international conference, ICMC 2017 (pp. 44-53). Springer Singapore.
[44] Habibi O, Chemmakha M, Lazaar M. Performance evaluation of CNN and pre-trained models for malware classification. Arabian Journal for Science and Engineering. 2023; 48(8):10355-69.
[45] Hindy H, Atkinson R, Tachtatzis C, Colin JN, Bayne E, Bellekens X. Utilising deep learning techniques for effective zero-day attack detection. Electronics. 2020; 9(10):1-16.
[46] Yin C, Zhu Y, Fei J, He X. A deep learning approach for intrusion detection using recurrent neural networks. IEEE Access. 2017; 5:21954-61.
[47] Swathy AM, Padmavathi G. Zero-day attack path identification using probabilistic and graph approach based back propagation neural network in cloud. Mathematical Statistician and Engineering Applications. 2022; 71(3s2):1091-106.
[48] Dhanya KA, Vajipayajula S, Srinivasan K, Tibrewal A, Kumar TS, Kumar TG. Detection of network attacks using machine learning and deep learning models. Procedia Computer Science. 2023; 218:57-66.
[49] Akshaya S, Padmavathi G. Enhancing zero-day attack prediction a hybrid game theory approach with neural networks. International Journal of Intelligent Systems and Applications in Engineering. 2024; 12:643-63.
[50] https://www.kaggle.com/datasets/kaggleprollc/nsl-kdd99-dataset/data. Accessed 20 February 2025.
[51] https://www.kaggle.com/code/mkashifn/celosia-zero-day-attack-detection-demo/input. Accessed 20 February 2025.
[52] Tavallaee M, Bagheri E, Lu W, Ghorbani AA. A detailed analysis of the KDD CUP 99 data set. In symposium on computational intelligence for security and défense applications 2009 (pp. 1-6). IEEE.
[53] Shaikh A, Gupta P. Real-time intrusion detection based on residual learning through ResNet algorithm. International Journal of System Assurance Engineering and Management. 2022:1-5.
[54] Haeser G, Ramos A. Constraint qualifications for Karush–Kuhn–Tucker conditions in multiobjective optimization. Journal of Optimization Theory and Applications. 2020; 187:469-87.

Swathy Akshaya is a Research Scholar in Computer Science at Avinashilingam University. She has published research articles in reputed national and international conferences, journals, and book chapters. Her research interests primarily include Cloud Computing and Cybersecurity.
Email: akshayakulandaivel@gmail.com

Padmavathi. G is the Dean of the School of Physical Sciences and Computational Sciences and a Professor in the Department of Computer Science at Avinashilingam Institute for Home Science and Higher Education for Women (Deemed to be University), Coimbatore. She has over 33 years of teaching experience and 25 years of research experience. Her research interests include Cybersecurity, Wireless Communication, and Real-Time Systems.
Email: padmavathi.avinashilingam@gmail.com

Appendix I

S. No.	Abbreviation	Description
1	AE	Auto-Encoder
2	AO	Adam Optimization
3	AUC	Area Under the Curve
4	AUC-ROC	Area Under the Curve - Receiver Operating Characteristic
5	AWS	Amazon Web Services
6	Bi-LSTM	Bidirectional Long Short-Term Memory
7	BPNN	Backpropagation Neural Networks
8	CANN	Center And Nearest Neighbor
9	CNN	Convolutional Neural Network
10	CNN-LSTM	Convolutional Neural Network with Long Short-Term Memory
11	CPU	Central Processing Unit
12	CSV	Comma-Separated Values
13	DCNN	Deep Convolutional Neural Network
14	DC-nZDASN	Deep Convolutional Zero-Day Adversarial Safety Network
15	DL	Deep Learning
16	DoS	Denial-of-Service
17	DDoS	Distributed Denial-of-Service
18	DT	Decision Tree
19	DTR	Decision Tree Regression
20	ECP	Energy Consumption Prediction

21	FL	Federated Learning
22	FP	False Positives
23	GAN	Generative Adversarial Network
24	GCP	Google Cloud Platform
25	GNB	Gaussian Naïve Bayes
26	GPU	Graphics Processing Unit
27	GS	Grid Search
28	GT	Game Theory
29	HGT	Hybrid Game Theory
30	IDS	Intrusion Detection Systems
31	IoT	Internet of Things
32	IMAP	Internet Message Access Protocol
33	LR	Logistic Regression
34	LSTM	Long Short-Term Memory
35	ML	Machine Learning
36	MLE	Maximum Likelihood Estimates
37	MSE	Mean Squared Error
38	NN	Neural Network
39	NSL- KDD	National Science Library- Knowledge Discovery In Databases
40	OS	Operating System
41	RF	Random Forest
42	ReLU	Rectified Linear Unit
43	R2L	Remote-to-Local
44	ResNet50	Residual Network
45	RNN	Recurrent Neural Network
46	RS	Random Search
47	SEC	Stacking Ensemble Classification
48	SEMG	Surface Electromyography
49	SMOTE	Synthetic Minority Over-sampling Technique
50	SV	Support Vector
51	SVM	Support Vector Machine
52	TCN	Temporal Convolutional Networks
53	SQL	Structured Query Language
54	U2R	User-to-Root
55	ZDA	Zero-Day Attack
56	ZSL	Zero-Shot Learning
57	IOB	In-of-Bag