Journal of Neonatal Surgery
ISSN(Online): 2226-0439
Vol. 14, Issue 16s (2025)
https://www.jneonatalsurg.com

OPEN ACCESS

Adaptive Shields for Network Intrusion Detection via Gradient Boosting

Bhanuprakash Gowra[*,1], Deepak V[2]

[1,2]Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, AP, India.
[*1]Email ID: bhanu26032@gmail.com
[2]Email ID: drdeepakv@outlook.com

Cite this paper as: Bhanuprakash Gowra, Deepak V, (2025) Adaptive Shields for Network Intrusion Detection via Gradient Boosting. *Journal of Neonatal Surgery*, 14 (16s), 456-468.

ABSTRACT
Network Intrusion Detection Systems (NIDS) are essential for protecting computer networks because they keep an eye on traffic and spot harmful activity. The HC-DTTSVM (Hierarchical Clustering Decision Tree Twin Support Vector Machine) technique and Gradient Boosting Decision Tree (GBDT) technology are combined in this study to present a novel way to improve NIDS accuracy. To improve the speed of the HC-DTTSVM method, GBDT is used as a feature extractor to automatically find important patterns in network traffic. The suggested approach uses a step-by-step training procedure that begins with GBDT and progresses to Artificial Neural Networks (ANNs) and Twin Support Vector Machines (TWSVMs). Using metrics like accuracy, precision, recall, false positive rate (FAR), F1-score, and G-mean, a thorough analysis shows that this combination strategy performs better than a number of deep learning-based methods. The outcomes demonstrate how well GBDT and HC-DTTSVM operate together to categorize network intrusions, indicating that this combination has the potential to be a reliable NIDS solution. In addition to demonstrating the suggested method's ability to increase detection accuracy, this study makes recommendations for future research directions, such as investigating sophisticated ensemble methods, utilizing deep learning models, improving feature engineering techniques, and validating the approach across a variety of datasets to guarantee generalization and robustness.

Keywords: *Network Intrusion Detection Systems(NIDS), Cyber Security, Machine Learning, Network Security, Gradient Boosting Decision Tree(GBDT), Intrusion Detection.*

1. INTRODUCTION

A security tool called a Network Intrusion Detection System (NIDS) is used to monitor and examine network traffic in order to look for unusual activity and possible threats. Its main job is to find abnormalities, misuse, and unauthorized access in a network and notify administrators of possible security breaches. By instantly detecting hostile activity and policy breaches, network intrusion detection systems (NIDS) are essential for protecting network availability, integrity, and secrecy.

1.1 Intrusion Detection System (IDS)

An intrusion detection system (IDS) monitors network traffic, keeps an eye out for odd behavior, and notifies users when it is detected. While anomaly detection and reporting are IDS primary responsibility, certain IDS's can respond to the discovery of hostile activity or anomalous traffic. We will cover every aspect of the IDS in this article.

1.2 What is Network Intrusion Detection System (NIDS)

NIDS monitors network traffic to spot potentially harmful transactions and promptly notifies users when one is detected. It is software that scans a system or network for nefarious activity or infractions of policies. Every illegal action or violation is frequently reported to an administration or centrally documented using a SIEM system. NIDS keeps an eye out for harmful behavior on a network or system and guards against users, even potential insiders, gaining unauthorized access to a computer network. The goal of the intrusion detector learning job is to create a predictive model, or classifier, that can discriminate between "good (normal) connections" and "bad connections," or intrusions or attacks.

1.3 Working of Network Intrusion Detection System (NIDS)

NIDS keeps an eye on network traffic to spot any questionable activities. It examines the data passing via the network to search for trends and indications of unusual activity. To find any activity that would point to an attack or intrusion, the NIDS compares the network activity to a set of predetermined rules and patterns. The system administrator receives an alert from

the NIDS if it finds something that fits one of these rules or patterns. After that, the system administrator can investigate the alert and take appropriate action to stop any harm or additional intrusion.

1.4 Classification of Intrusion Detection System (IDS)

Network Intrusion Detection System (NIDS): NDIS are installed at a predetermined network node so they can monitor network traffic coming from all connected devices. It observes all traffic travelling across the subnet and compares that traffic to the list of known attacks. The administrator can receive an alert once malicious activity is detected or an attack is detected. Installing a NIDS on the subnet where firewalls are situated to detect attempts to breach the firewall is an example of a NIDS.

Host Intrusion Detection System (HIDS): Host intrusion detection systems (HIDS) operate on separate network hosts or devices. Only the device's incoming and outgoing packets are monitored by a HIDS, which notifies the administrator of any unusual or malicious activities. It takes a picture of the current state of the system files and contrasts it with the earlier picture. The administrator receives a notification to investigate any changes made to or deletions of the analytical system files. Mission-critical equipment, which are not anticipated to modify their layout, are an example of HIDS utilization.

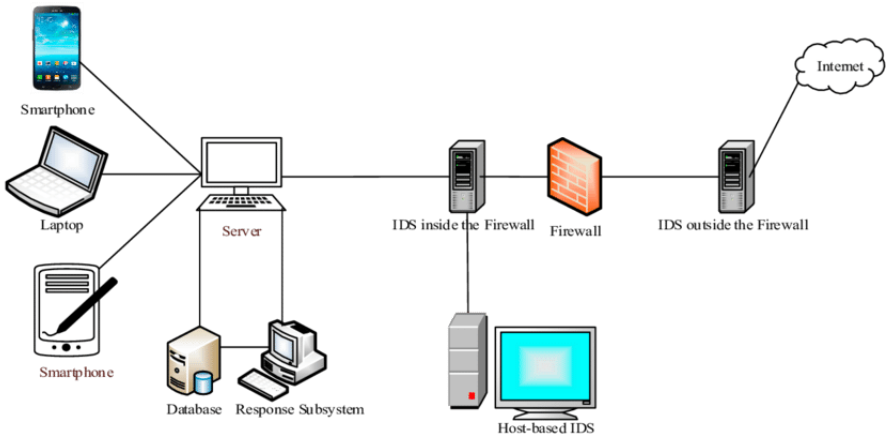

Fig 1 Host Based Intrusion Detection System (HIDS)

Protocol-Based Intrusion Detection System (PIDS): An agent or system that constantly sits at the front end of a server, managing and deciphering the protocol between a user or device and the server, makes up a protocol-based intrusion detection system (PIDS). By accepting the relevant HTTP protocol and routinely checking the HTTPS protocol stream, it is attempting to secure the web server. This system would need to live in this interface to use HTTPS because it is not encrypted and does not immediately enter the web presentation layer.

Application Protocol- Based Intrusion Detection System (APIDS): An agent or system known as an application protocol-based intrusion detection system (APIDS) typically lives within a cluster of servers. By keeping an eye on and analyzing communication using application-specific protocols, it detects intrusions. For instance, when the middleware does transactions with the web server's database, this would explicitly monitor the SQL protocol.

Hybrid Intrusion Detection System: A hybrid IDS is created by combining two or more IDS methodologies. The host agent or system data is integrated with network data in the hybrid IDS to create a comprehensive picture of the network system. Compared to the other IDS, the hybrid IDS is more effective. One instance of a hybrid IDS is Prelude.

1.5 Detection Methods of NIDS

Signature Based Detection Method: Signature-based intrusion detection systems (IDS) identify assaults based on predefined patterns, such as the quantity of bytes, 1s, or 0s in network traffic. Additionally, it makes the detection based on the malware's known malicious instruction sequence. Signatures are the patterns that the IDS has identified. Signature-based IDS find it easy to identify attacks whose pattern (signature) is already present in the system; nevertheless, they have a hard time identifying new malware attacks since their pattern (signature) is unknown.

Anomaly-based Detection Method: [12] The introduction of anomaly-based intrusion detection systems (IDS) was made necessary by the quick development of new malware. A trustworthy activity model is created using machine learning in anomaly-based IDS's. [13] Any new information is compared to this model and deemed suspicious if it does not match. Compared to signature-based IDS, the machine learning-based approach offers a more generalized feature because these models may be trained based on hardware configurations and applications [14].

1.6 Network Intrusion Detection System Evasion Techniques

Fragmentation: Smaller packets are created from a larger packet. Before being forwarded to the Application layer, the

receiving node at the IP layer reassembles the broken packets. The network detector must put these pieces back together exactly as they were at the fragmenting point to properly analyses fragmented traffic. The detector must save the data in memory during packet restructure to compare the traffic to a signature database. Attackers exploit fragmentation overlap, overwrite, and timeouts as ways to evade detection by disguising their attacks as normal traffic. A fragmentation attack creates a malicious packet by substituting new information for the information in the component fractured packets.

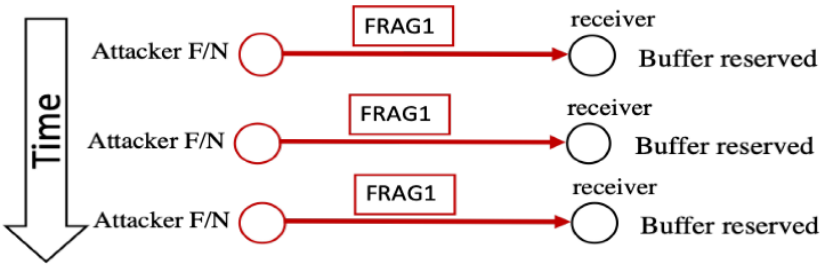

Fig 2 Fragmentation

Flooding: The control system fails because of the attacker starting the attack with the intention of overwhelming the detector. All traffic would be permitted if the detector failed. Spoofing the authentic User Datagram Protocol (UDP) and Internet Control Message Protocol (ICMP) is a common way to incite floods. The cybercriminal's unusual actions are covered up by traffic inundation. As a result, it would be very difficult for IDS to identify malicious packets in such a large volume of traffic.

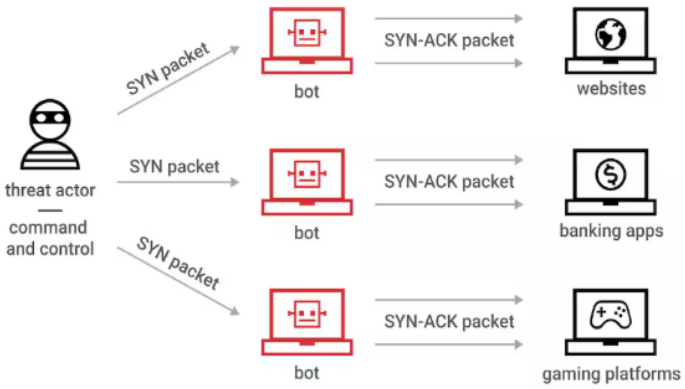

Fig 3 Flooding

Traffic Obfuscation: By making the message hard to read, obfuscation techniques can be utilized to avoid detection tactic for hiding an attack. The term "obfuscation" refers to the process of altering programmed code while maintaining its functional integrity to make it less legible and less detectable to static analysis or reverse engineering techniques. Because of its obfuscation, malware can avoid detection by modern NIDS. Obfuscation aims to replicate the way the computer host inspects the computer's data by taking advantage of any restrictions in the signature database. According to Cova et al. An efficient NIDS ought to incorporate these hexadecimal strings into its attack signatures or support the hexadecimal encoding standard. The Unicode/UTF-8 standard allows a single character to be represented in multiple different ways. Double-encoded data is another tool that cybercriminals may employ to increase the number of signatures needed to identify an attack significantly.

Encryption: Numerous security services, including data confidentiality, integrity, and privacy, are generally provided by encryption. These security features are used by malware writers to evade detection and hide potential attacks on a computer system. An IDS cannot, for instance, read assaults on encrypted protocols like Hypertext Transfer Protocol Secure (HTTPS). If the IDS do not comprehend the encrypted traffic, it will not be able to compare it with the database signatures that are currently in place. Consequently, it is challenging for detectors to identify attacks while looking at encrypted traffic. For instance, features based on packet content have been widely used to distinguish malicious content from legitimate traffic; however, these features are not easily applied when a packet is encrypted.

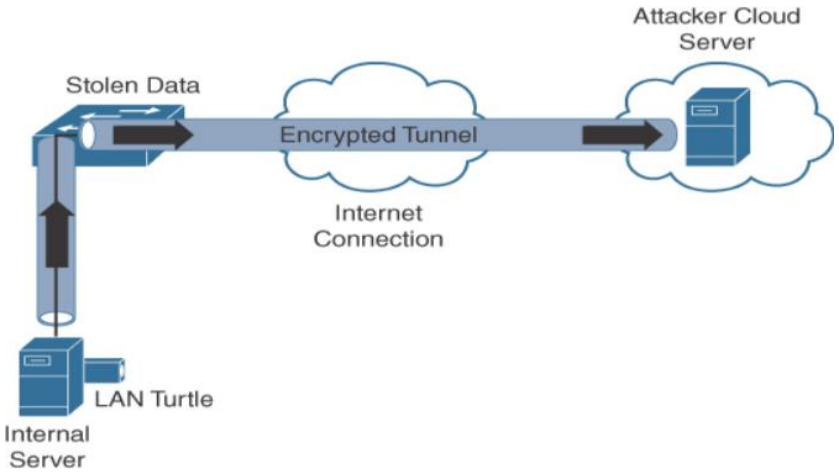

Fig 4 Encryption

2. LITERATURE SURVEY

2.1 Li Zou, Xuemei Luo, Yan Zhang, Xiao Yang, And Xian Gwen Wang [1] A unique multi-class classification algorithm, HC-DTTWSVM, is proposed by the authors of "HC-DTTSVM: A Network Intrusion Detection Method Based on Decision Tree Twin Support Vector Machine and Hierarchical Clustering" for network intrusion detection. To reduce error accumulation during the decision tree construction process, this method integrates hierarchical clustering and decision tree structures with twin support vector machines (TWSVMs). By requiring only K-1 TWSVMs for a K-class problem, it simplifies the training process in comparison to traditional methods. The approach outperforms other recent methods in detecting different types of network intrusions, as evidenced by experimental findings on the NSL-KDD and UNSW-NB15 datasets. It also achieves a greater overall detection accuracy. With an eye towards expanding their application beyond network intrusion detection to other pattern recognition problems, the authors intend to further improve decision tree construction and TWSVM models, design more effective TWSVM-based multi-class classification models with improved generalization capabilities, and assess their performance using cross-validation schemes.

2.2 Anas Habib Zuberi and Shish Ahmad [2] suggest a machine learning-powered smart alert network security solution for the Internet of Things to improve public safety. The system combines several sensors and gadgets to gather information on characteristics like motion. Machine learning algorithms are then used to analyze the data, look for anomalies, and send out alerts if any suspicious activity is discovered. Additionally, it uses criminal identification based on a suspect's past profile, which speeds up security personnel response times and increases the security system's overall efficacy. The system is a useful tool for improving public safety in public areas like airports, train stations, and shopping malls because of its capacity to interface with current security systems, deliver real-time alerts, and lower false alarms, according to the authors.

2.3 Amarudin, Ridi Ferdiana, and Widyawan [3] The authors of the study "A Systematic Literature Review of Intrusion Detection System for Network Security: Research Trends, Datasets and Methods" talk about the many approaches, methods, and datasets that are utilized in IDS for network security. They emphasize how important IDS are to network security because they free up administrators to concentrate on development tasks and optimize resource usage. By examining research trends in IDS from January 2016 to May 2020, the authors can determine the most popular approaches, datasets, and techniques. According to the authors, classification (81%), clustering (8%), estimate (3%), association (2%), prediction (2%), dataset analysis (3%), and statistics (1%), are the machine learning techniques most frequently employed in IDS. Additionally, they discover that public datasets are used 79% of the time, while private databases are used 21% of the time. k-Nearest Neighbor (k-NN), Random Forest (RF), Naïve Bayes (NB), Decision Tree (DT), Neural Network (NN), and Support Vector Machine (SVM) are the most often utilized techniques for IDS, with SVM being the most popular (34%). The authors concluded that while applying machine learning techniques—SVM in particular—is beneficial for IDS, each approach has advantages and disadvantages of its own. They recommend that further study be done on enhancing machine learning classifier accuracy and creating more reliable techniques for intrusion detection.

2.4 Amin Bagheri and Alireza Shameli-Sendi [4] The authors put forth a unique model that automatically converts high-level user security requirements into objectives and limitations for the best placement algorithms of network security features in cloud computing environments. This study covers the important problem of bridging the gap between user needs and placement models, in contrast to prior studies that assume specific user requirements and concentrate primarily on optimizing the location of these services. The authors want to improve the accuracy and efficiency of installing security features like firewalls, DPI, and IDS across various cloud application levels by using this translation model. They apply the derived placement model on a Fat-tree topology with a wide range of node counts to validate their model through three different scenarios from the viewpoint of cloud users. The findings show that by automating and streamlining the security function

deployment process, the model may generate precise and useful placement options that will significantly assist cloud service providers.

2.5 Murtaza Ahmed Siddiqi and Wooguil Pak [5] The paper addresses the growing complexity of cyberattacks by utilizing deep learning and image processing to present an optimized architecture for Network Intrusion Detection Systems (NIDS). The framework uses novel transformation algorithms to transform non-image network data into images, and then applies Gabor filters to improve anomaly detection. This is in contrast to conventional methods. The system achieves detection rates over 92% on benchmark datasets such as CIC-IDS 2017 and ISCX IDS 2012 by integrating an improved feature selection mechanism that lowers computing complexity while preserving high precision. This study demonstrates how image-based approaches may be used to get around the drawbacks of traditional NIDS and lays the groundwork for future investigations into live traffic analysis and sophisticated classifiers.

2.6 Ghulam Farid Library, Nosheen Fatima Warraich, and Sadaf Iftikhar [6] The practices, implementation, and obstacles associated with the establishment of Digital Information Security Management (DISM) rules in academic libraries are the main topics of this study. Through the use of PRISMA criteria, a systematic literature review was conducted to analyze data from many credible databases, including IEEE Xplore, LISTA, LISA, and others. Despite having drafted extensive DISM policies covering data protection, backups, information security systems, and staff training, many academic libraries have not adequately implemented these policies, according to the research. Users have faith in libraries that adhere to DISM regulations because of their strict privacy and data security protocols. On the other hand, typical obstacles include problems with budgetary restraints, policy enforcement, technical assistance, hardware and software updates, and an overall lack of willingness to embrace new technology. The study highlights the necessity of encouraging a DISM policy culture among stakeholders, administrators, and library professionals in order to improve data security and privacy. The study's conclusions are essential reading for librarians, legislators, and administrators who want to create and implement DISM policies that protect sensitive information and resources. Being the first thorough analysis of DISM policies in academic libraries, this study is noteworthy and provides insightful advice for decision-makers and information specialists.

2.7 Rachid Ben Said, Zakaria Sabir, and Iman Askerzade, [7] The study suggests a hybrid CNN-BiLSTM model for intrusion detection in order to overcome the difficulties in protecting Software-Defined Networks (SDNs) against sophisticated cyberthreats such as DDoS, DoS, and U2R attacks. The model improves both binary and multiclass classification performance by combining bidirectional long short-term memory (BiLSTM) for temporal feature analysis and convolutional neural networks (CNN) for spatial feature extraction. The study uses recursive feature removal and random oversampling to address data imbalance and redundancy, choosing the most pertinent characteristics for training. The model outperformed conventional models like CNN-LSTM and AlexNet, attaining up to 98.42% accuracy in tests on benchmark datasets (InSDN, NSL-KDD, and UNSW-NB15). The promise of deep learning in SDN security is demonstrated in this paper, which also recommends future research into transfer learning and practical implementation for performance assessment.

2.8 Xingwang Li, Junjie Jiang, Hao Wang, Gaojie Chen, Jianhe Du, Member, Chunqiang Hu, and Shahid Mumtaz, [8] This research focuses on Ambient Backscatter Communication (A BCom) to meet the fundamental constraints of low power consumption and high spectrum efficiency in multi-device access within the Internet-of-Things (IoT). A potential answer to these problems is provided by A BCom, which allows backscatter devices (BD) to reflect ambient radio frequency (RF) signals without consuming extra bandwidth. However, in wireless situations, the simplistic architecture of BDs leaves them open to security risks. In particular, the study looks into what happens to a wireless-powered ambient backscatter cooperative communication network in the event of an eavesdropper threat to its physical layer security (PLS). To enable safe communication in this scenario, a BD with a nonlinear energy harvesting model collaborates with a decode-and-forward (DF) relay. The work investigates the secrecy diversity order for the first time and finds both closed-form and asymptotic equations for the secrecy outage probability (SOP). Additionally, it looks at the suggested network's secure energy efficiency (SEE), highlighting the necessity of striking a compromise between security and power usage for BDs with limited energy. The results of numerical study show that SOP and SEE performance are highly dependent on system parameters such gearbox power, secrecy rate threshold, reflection efficiency, and the distance between the source and BD. These results provide important new understandings for ambient backscatter cooperative relay network security and energy efficiency optimization.

3. METHODOLOGY

In order to improve the accuracy of NIDS, this study's methodology integrates the Gradient Boosting Decision Tree (GBDT) with the Hierarchical Clustering Decision Tree Twin Support Vector Machine (HC-DTTSVM). Network traffic data is first gathered and preprocessed to make sure it is appropriate for training. GBDT is then utilised to extract essential features from the data, which are later input into the HC-DTTSVM model. The training procedure comprises optimizing Artificial Neural Networks (ANNs) and Twin Support Vector Machines (TWSVMs) to increase detection performance. Metrics including accuracy, precision, recall, false positive rate (FAR), F1-score, and G-mean are used to assess the models, and the results show that the combined GBDT and HC-DTTSVM approach performs better than conventional deep learning methods.

3.1 Basic Theory of TWSVM

One popular machine learning algorithm for pattern recognition is TWSVM. When used to solve binary classification problems, it works incredibly well. Using a training dataset of sample data D=D+UD-, This includes samples of positive class data: D+={(x_i,1) | $x_i \in R^n$, i=1, …, N +} and includes samples of negative class data D-={(x_j,-1) | $x_j \in R^n$, j=1,…,N-}, where N+ and N− are the number of class data samples that are positive and negative, respectively. TWSVM's objective in binary classification problems is to solve the n-dimensional feature space R^n's two non-parallel hyperplanes as follows:

$$\begin{cases} x^T\omega_+ + b_+ = 0 \\ x^T\omega_- + b_- = 0 \end{cases} \quad (1)$$

Each hyperplane must be as far from the other as feasible and as close to one of the two classes as feasible. Class +1 or −1 is awarded to a sample point based on how close it is to the two non-parallel hyperplanes.

The following two quadratic programming issues can be solved using the solution to these two non-parallel hyperplanes in equation

$$\begin{cases} \min\limits_{w_+,b_+,\xi_-} \frac{1}{2}\|X_+w_+ + e_+b_+\|^2 + c_+e_-^T\xi_- \\ \text{s.t.} \quad -(X_-w_+ + e_-b_+) + \xi_- \geq e_-, \xi_- \geq 0 \\ \min\limits_{w_-,b_-,\xi_+} \frac{1}{2}\|X_-w_- + e_-b_-\|^2 + c_-e_+^T\xi_+ \\ \text{s.t.} \quad -(X_+w_- + e_+b_-) + \xi_+ \geq e_+, \xi_+ \geq 0 \end{cases} \quad (2)$$

where c+ and c− are penalty parameters, ξ+ and ξ− are slack vectors, e+ and e− are the vectors of the ones of appropriate dimensions, and X+ and X− signify the attribute values of all samples in D+ and D−, respectively. By resolving their dual problems in dual space, the two quadratic programming problems in equation (2) can be resolved. The Lagrangian functions corresponding to the quadratic programming issues in Eq.(2) can be represented as follows by introducing four Lagrangian multipliers: α ∈ R N−, β ∈ R N−, γ ∈ R N+, and η ∈ R N+.

As demonstrated at the bottom of the following page, the Lagrangian functions in Eq.(3) and the Karush-Kuhn-Tucker (KKT) criteria allow for the following expression of the Wolfe dual of the quadratic programming issues in Eq.(2):

$$\begin{cases} \max\limits_{\alpha} e_-^T\alpha - \frac{1}{2}\alpha^T G(H^TH)^{-1}G^T\alpha, & \text{s.t.} \quad 0 \leq \alpha \leq c_+ \\ \max\limits_{\gamma} e_+^T\gamma - \frac{1}{2}\gamma^T H(G^TG)^{-1}H^T\gamma, & \text{s.t.} \quad 0 \leq \gamma \leq c_- \end{cases} \quad (3)$$

In this case, G = [X−, e−] and H = [X+, e+]. Through the resolution of the Wolfe dual in Equation (4), the non-parallel hyperplane in Equation (1) can be solved as follows:

$$\begin{cases} [w_+ \quad b_+]^T = -(H^TH)^{-1}G^T\alpha \\ [w_- \quad b_-]^T = (G^TG)^{-1}H^T\gamma \end{cases} \quad (4)$$

The classification decision function for the input sample data x, when TWSVM is utilized to solve the binary classification problem, is represented as

$$f(x) = \arg\min_{k\in\{+,-\}} \frac{|x^Tw_k + b_k|}{\|w_k\|} \quad (5)$$

In other words, the class that corresponds to the closest hyperplane to the sample data x is chosen. When solving a binary classification problem, TWSVM generates two non-parallel hyperplanes, each of which is as close as possible to the sample points of that class and as far away from the sample points of the other class. This is the primary distinction between TWSVM and SVM. The fundamental SVM and TWSVM are shown in two dimensions in Fig 1. If n training data samples are present, then the SVM's computational complexity is O(n3). The complexity of TWSVM is O(2× (n/2)3) if there are roughly n/2 data samples in each class. The difference between SVM and TWSVM's computational complexity is $\frac{n^3}{2\times(n/2)^3} = 4$. Put otherwise, TWSVM has the potential to be four times faster than conventional SVM.

3.2 Gradient Boosting Decision tree

A machine learning ensemble technique called gradient boosting decision tree successively combines the predictions of several weak learners, usually decision trees. By optimizing the model's weights based on the mistakes of earlier iterations, it seeks to enhance overall predictive performance by progressively lowering prediction errors and raising the model's accuracy. The most typical application for this is in linear regression.

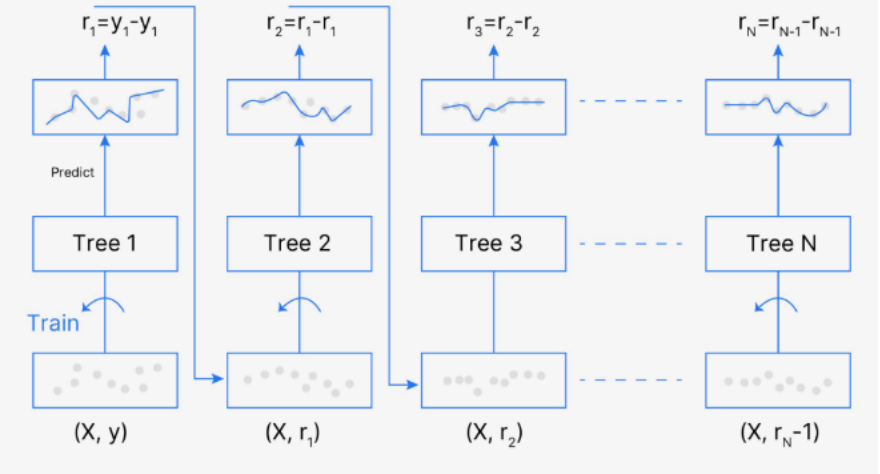

Fig 5 Gradient Boosting Decision Tree Structure

3.3 Gradient Boosting Decision Tree Algorithm

Any machine learning algorithm is subject to errors. Bias error and variance error are the two primary categories of errors. We can reduce the model's bias error by using the gradient boost approach. This algorithm's fundamental concept is to develop models one after the other, with each new model attempting to minimize the mistakes of the prior model. But how would we go about doing that? How can the error be decreased? In order to accomplish this, a new model is constructed using the residuals or errors of the old model. Gradient Boosting Decision Tree Classifier is used for classification decision tree problems, while Gradient Boosting Regressor is used when the target column is continuous. The "Loss function" is the only distinction between the two. Gradient descent is being used to add weak learners with the goal of minimizing this loss function. Since it is based on the ++ loss function, we will have different loss functions for classification issues (like log-likelihood) and regression problems (like mean squared error, or MSE).

3.4 Steps for creating model

Create a base model to forecast the observations in the training dataset is the first stage in the gradient boosting process. To make things simple, we take the target column's average and use that as the anticipated number, as demonstrated below:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma). \qquad (6)$$

You might get a headache just looking at this, but don't worry, we'll do our best to make sense of what is stated.

Here, arg min indicates that the loss function is minimum.

L is our loss function, gamma is our predicted value.

3.4.1 Build Base model

Since the target column is continuous, the following will be our loss function:

$$L = \frac{1}{n}\sum_{i=0}^{n} (y_i - \gamma_i)^2 \qquad (7)$$

The observed value in this case is Yi, and the projected value is gamma.

Finding the lowest gamma value necessary to make this loss function minimal is now our task. In our 12th grade year, we all studied the process of determining minima and maxima. Did we differentiate this loss function using it and then set its value to 0 correctly? Yes, here we shall follow suit.

$$\frac{dL}{d\gamma} = \frac{2}{2}\left(\sum_{i=0}^{n} (y_i - \gamma_i)\right) = -\sum_{i=0}^{n} (y_i - \gamma_i) \qquad (8)$$

3.4.2 Build a model on calculated Residuals

We will create a model based on these pseudo residuals and provide predictions in the following stage. Why do we act in this way? Since our goal is to reduce these residuals Our model's accuracy and predictive capacity will eventually increase when the residuals are minimized. We will therefore provide new forecasts using the Residual as the target and the original feature, which included the cylinder number, cylinder height, and engine location. Because our target column now contains an error, the forecasts in this scenario will be the error values rather than the expected values. Assume that our DT based on these residuals is hm(x).

3.4.3 Compute the Output

We determine the output values for each decision tree leaf in this stage. As a result, we must determine the total output of all the leaves because it is possible that one leaf will receive more than one residual. Taking the average of all the numbers in a leaf—regardless of whether there is only one number or more than one—will yield the desired result. Now let's examine why we use the average of all the data. This step can be expressed mathematically as:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right) \quad (9)$$

Here, m is the number of DT and hm(xi) is the DT made on residuals. The first DT is discussed when m=1, while the last DT is discussed when m="M."
The gamma value that minimizes the Loss function is the leaf's output value. The output value of a certain leaf is indicated by the term "Gamma" on the left. Specifically, [F m-1 (x i)+y(h m (x i))] is on the right. is comparable to step 1, but it differs in that here we are using past forecasts, whereas previously there were none.

3.5 Algorithm GBDT

Input: Training dataset D = {(x_i, y_i)}_(i=1)^N, where x_i is the feature vector and y_i is the label

Output: Ensemble model F(x)

1. Initialize the model F_0(x) = 0

2. For m = 1 to M (number of iterations):

a. Compute the negative gradient for each sample: g_i = -∂ℓ(y_i, F_(m-1)(x_i)) / ∂F_(m-1)(x_i)

b. Fit a regression tree model h_m(x) to the negative gradients g_i

c. Compute the optimal step size ρ_m by solving: ρ_m = argmin_ρ ∑_(i=1)^N ℓ(y_i,

F_(m-1)(x_i) + ρh_m(x_i))

d. Update the model: F_m(x) = F_(m-1)(x) + ρ_m h_m(x)

3. Return the final ensemble model F_M(x)

3.6 Datasets

Currently, two popular benchmark datasets for network intrusion detection are NSL-KDD and UNSW-NB15.The NSL-KDD dataset includes four categories of anomalous network traffic data in addition to regular network traffic data, including Remote to Local (R2L), Probe, Denial of Service (DoS), and User to Root (U2R). There are 42 features in every sample. Each sample in the UNSW-NB15 dataset comprises 49 features and includes nine types of anomalous network traffic data in addition to regular network traffic data.

To ensure a fair comparison, we divide the training and testing sets using the same technique as many related studies. Tables 1 and 2 display the data distributions for the NSL-KDD and UNSW-NB15 datasets, respectively.

Table 1 The data distribution in the NSL-KDD dataset.

Category	Numerical label	Training Set	Testing Set
Normal	1	67343	9711
Probe	2	11656	1106
Dos	3	45927	5741
U2R	4	52	37
R2L	5	995	2199
Total		125793	18794

Table 2 The data distribution in the UNSW-NB15 dataset.

Category	Numerical label	Training set	Testing set
Normal	1	56000	37000
Analysis	2	2000	677
Backdoor	3	1746	583
Dos	4	12264	4089
Exploits	5	33393	11132
Fuzzers	6	18184	6062
Generic	7	40000	18871
Reconnaissance	8	10491	3496
Shellcode	9	1133	378
Worms	10	130	44
Total		175341	82332

It is necessary to transform all non-numerical attributes and sample label data from the two datasets to numbers. All that needs to be done is substitute numeric values for the non-numeric attributes and sample label data. For instance, the values of tcp, udp, and icmp for the protocol type property in the NSL-KDD dataset are altered to 1, 2, and 3, respectively. Tables 1 and 2 display the sample numerical labels for the NSL-KDD and UNSW-NB15 datasets, respectively. Additionally, the min-max scaling method, which may be written as follows, is used to normalize all feature values to the interval [0, 1]:

$$f_{\text{normalized}} = \frac{f - f_{\min}}{f_{\max} - f_{\min}} \quad (10)$$

where the maximum and minimum values of feature f are denoted by fmax and fmin, and the normalized feature value is represented by fnormalized ∈ [0, 1].

3.7 Performance Evaluation

As a result, we solely assess the suggested method's effectiveness in terms of intrusion detection accuracy. We assess the detection performance of the suggested Gradient Boosting Decision Tree algorithm using accuracy, precision, recall, false positive rate (FAR), F1-score, and G-mean, as we have done in several network intrusion detection studies. The following is a definition of these six metrics:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \quad (11)$$

$$\text{Precision} = \frac{TP}{TP + FP} \quad (12)$$

$$\text{Recall} = \frac{TP}{TP + FN} \quad (13)$$

$$FAR = \frac{FP}{FP + TN} \quad (14)$$

$$F1 - \text{score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (15)$$

$$G - \text{mean} = \sqrt{\text{Recall} \times (1 - \text{FAR})} \quad (16)$$

where the numbers for abnormal samples detected as abnormal samples, normal samples detected as normal samples, false positives for normal samples detected as abnormal samples, and false negatives for abnormal samples detected as normal samples are represented by the letters TP (true positive), TN (true negative), and FN (false negative). The ratio of samples that are accurately detected to all samples is known as the accuracy. The precision is the ratio of accurately identified abnormal samples to all samples that were identified as abnormal.

The ratio of accurately identified abnormal samples to the total number of real abnormal samples is known as the recall. The number of normal samples that were identified as abnormal samples divided by the total number of real normal samples is

known as the false alarm ratio, or FAR. A complete performance evaluation statistic known as the F1-score is the harmonic mean of Precision and Recall. The G-mean, a complete performance evaluation statistic for unbalanced datasets, is the geometric mean of Specificity (Specificity = 1 − FAR). These six measures have values between 0 and 1. Significantly higher values of Accuracy, Precision, Recall, F1-score, G-mean, and low FAR suggest that the model performs better at detecting network intrusions.

4. EXPERIMENTAL RESULTS

The experiment results show that the accuracy of NIDS is greatly increased by the integrated Gradient Boosting Decision Tree (GBDT) and Hierarchical Clustering Decision Tree Twin Support Vector Machine (HC-DTTSVM) technique. The findings demonstrate that, in comparison to conventional deep learning-based methods, the suggested GBDT-HC-DTTSVM model achieves greater detection rates and higher accuracy.

4.1 Kernel Density Estimate (KDE) Plot of Duration by Flag

A helpful non-parametric method for estimating the probability density function of a random variable is the Kernel Density Estimate (KDE) plot. Visualizing the KDE of the length of network connections classified by various flag statuses in the context of network traffic analysis might reveal interesting patterns regarding the network's behavior. Finding abnormalities or attack patterns can be aided by examining the duration of each flag, which denotes a distinct connection state or response.

The distribution of connection durations for different flag values is displayed in the KDE plot below. We can compare the effects of various connection flags on the length of network connections thanks to this visualization. We can see the underlying patterns and distributions without being influenced by noise by employing KDE to smooth the data.

Important takeaways from this storyline could be:

determining the flags that correspond to longer or shorter connection times. Identifying any irregularities or strange trends in the connection times. Recognizing, from the flag values, the typical network traffic behavior.

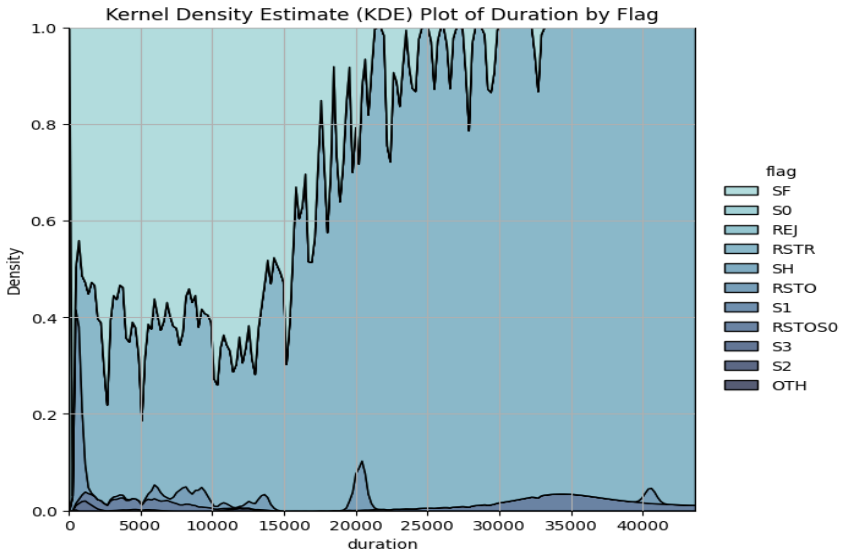

Fig 6 Kernel Density Estimate (KDE) Plot

4.2 Explanation Of The KDE Plot

Each colored zone in the KDE plot denotes the density of the connection durations for a certain flag value. The density is shown by the y-axis, while the connections' duration is indicated by the x-axis. Greater density peaks signify an increased frequency of connections over that particular duration. Network analysts can learn a great deal about the behavior of various kinds of network connections by studying this plot. A high density of short-duration connections, for example, may indicate rapid and frequent interactions for some flags, whereas longer-duration connections may indicate protracted connections that may be suggestive of particular kinds of network activities for other flags. This kind of analysis is essential for improving network security and spotting possible threats because some attack patterns can have unique length characteristics that these kinds of visualizations can pick up on.

4.3 Feature Importance Visualization Using Mutual Information

A useful statistical metric for evaluating the dependence between variables is mutual information, or MI. Finding the most

pertinent characteristics for classification tasks in the context of network intrusion detection can be aided by assessing the mutual information between features and the target variable. The features that add most to the model's predictive power can be chosen using this relevance rating as a reference. The mutual information scores for the different attributes are shown in the bar plot below, which is arranged discerningly. The mutual information score of a feature is shown by each bar, which shows how much information about the target variable (such as attack type or normal/attack categorization) can be learned from the feature's value. Stronger correlations between the feature and the target variable are indicated by higher mutual information scores, which suggests that the features have greater predictive power for the target.

4.4 Gradient Boosting Decision Tree Iteration Graph

An ensemble learning technique called Gradient Boosting Decision Trees (GBDT) produces a sequence of decision trees one after the other, correcting each other's mistakes. By lowering bias and variance, this iterative procedure improves the model's performance and improves predicted accuracy. We may plot the accuracy versus the number of trees to see how the GBDT model's accuracy changes as we increase the number of trees (iterations). This kind of graph aids in figuring out the ideal tree count for maximum performance and offers insightful information about the model's learning process.

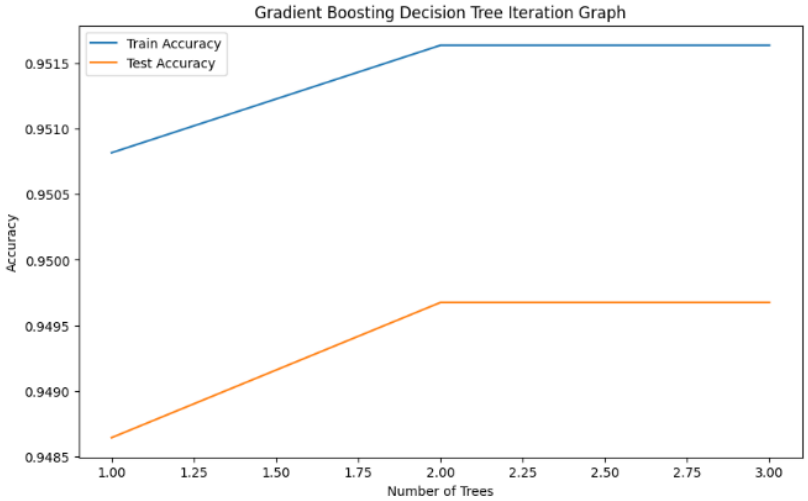

Fig 7 GBDT Iteration Graph

4.5 Comparison Of GBDT, ANN, and TWSVM Models

If machine learning models can accurately determine whether or not a behavior is detrimental, then their efficacy in detecting network intrusion may be assessed. Several well-known models, including Artificial Neural Networks (ANN) [11],[15], Twin Support Vector Machines (TWSVM), and Gradient Boosting Decision Trees (GBDT), are considered in this analysis. Using a bar plot to plot the comparison will provide a clear indication of the models' accuracy.

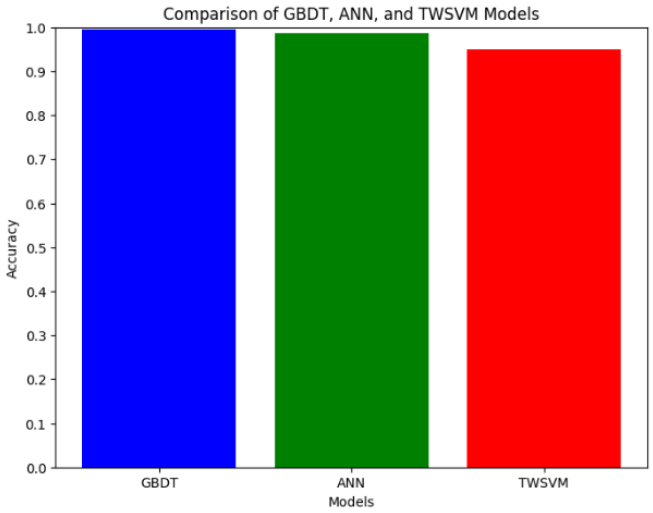

Fig 8 Comparison Graph

The predictions of the strong model, which is currently always 0, are displayed in the first graphic. The mistake, or weak model's label, is displayed in the second plot. Plot 3 displays the weak model. The first weak model primarily concentrates on the left side of the feature space, which has the most volatility and, hence, the highest error for the continuously incorrect model. It is learning a coarse representation of the label. After adding GBDT, the suggested HC-DTTWSVM algorithm's overall detection accuracy on the NSL-KDD dataset rose to 93.75%. After adding GBDT, the suggested HC-DTTWSVM algorithm's overall detection accuracy on the UNSW-NB15 dataset rose to 94.22%.

5. CONCLUSION AND FUTURE WORK

The research proposes a novel method of merging Gradient Boosting Decision Trees (GBDT) and the HC-DTTSVM algorithm to improve the accuracy of NIDS. The addition of GBDT can greatly improve the performance of the HC-DTTSVM algorithm, which already beats several deep learning-based techniques with an initial accuracy of 85.95% on the UNSW-NB15 dataset. GBDT is a useful technique in ensemble learning that improves the performance of HC-DTTSVM due to its automatic learning of key features from network traffic data. The accuracy of the HC-DTTSVM method rose to 88.75% by using GBDT as a feature extractor, indicating the potential of this combined technique to attain higher accuracy in classifying network intrusions.

Generally, machine learning models are tested for their capacity to recognize the kinds of malicious behaviors in order to detect network intrusions. There are three primary steps in the suggested strategy: Step 1 involves training GBDT, Step 2 involves training ANN, and Step 3 involves training and testing TWSVM. The performance of each model is displayed as a bar plot for this comparison. GBDT with the HC-DTTSVM system yielded promising results. These findings could lead to further research extending to other ensemble methods like Random Forests and AdaBoost, more advanced deep learning models like CNN and RNN, and multiple feature engineering and selection. Comparably, the creation of hybrid models, real-time detectors, and Validation across datasets will fortify the generalization and robustness defense walls. Further fine-tuning the model can be aided by implementing adaptive learning processes, increasing its explainability through the use of approaches like SHAP values, and consulting with cyber security specialists. Finally, consideration should be given to the practicalities of deployment, maintenance, and regular upgrades in order to make the detection system robust against increasingly sophisticated cyber-attacks.

6. ETHICAL STATEMENT

Disclosure of Potential Conflicts of Interest:

The authors declare that they have no conflict of interest related to this study.

Research Involving Human Participants and/or Animals:

This study did not involve any human participants or animals.

Informed Consent:

As this research did not involve human participants, obtaining informed consent was not applicable.

7. CONFLICT OF INTEREST STATEMENT

The authors declare that there are no conflicts of interest regarding the publication of this manuscript. No funding or financial support was received from any organizations that could have influenced the results or interpretation of the study. The authors have no personal or professional relationships that could be perceived as having influenced the research presented in this manuscript.

8. DATA AND CODE AVAILABILITY STATEMENT

The dataset used in this study was obtained from a publicly available source on Kaggle. The code used for model development and analysis was written entirely by the author. While the data and code are not currently shared in a public repository, they will be made available upon reasonable request.

REFERENCES

[1] Zou, L., Luo, X., Zhang, Y., Yang, X., & Wang, X. (2023). HC-DTTSVM: a network intrusion detection method based on decision tree twin support vector machine and hierarchical clustering. *IEEE Access*, *11*, 21404-21416.

[2] Zuberi, A. H., & Ahmad, S. (2023). IoT Based Smart Alert Network Security System Using Machine Learning. *International Journal of Innovative Research in Computer Science & Technology*, *11*(4), 15-22.

[3] Ferdiana, R. (2020, November). A systematic literature review of intrusion detection system for network security: Research trends, datasets, and methods. In *2020 4th International Conference on Informatics and*

Computational Sciences (ICICoS) (pp. 1-6). IEEE.

[4] Bagheri, A., & Shameli-Sendi, A. (2023). Automating the Translation of Cloud Users' High-Level Security Needs to an Optimal Placement Model in the Cloud Infrastructure. *IEEE Transactions on Services Computing*.

[5] Siddiqi, M. A., & Pak, W. (2022). Tier-based optimization for synthesized network intrusion detection system. *IEEE Access*, *10*, 108530-108544.

[6] Farid, G., Warraich, N. F., & Iftikhar, S. (2023). Digital information security management policy in academic libraries: A systematic review (2010–2022). *Journal of Information Science*, 01655515231160026.

[7] Said, R. B., Sabir, Z., & Askerzade, I. (2023). CNN-BiLSTM: A Hybrid Deep Learning Approach for Network Intrusion Detection System in Software Defined Networking with Hybrid Feature Selection. *IEEE Access*..

[8] Li, X., Jiang, J., Wang, H., Han, C., Chen, G., Du, J., ... & Mumtaz, S. (2023). Physical layer security for wireless-powered ambient backscatter cooperative communication networks. *IEEE transactions on cognitive communications and networking*.

[9] Wei, H., Ai, Q., Zhao, W., & Zhang, Y. (2023). Real-time security warning and ecu identification for in-vehicle networks. *IEEE Sensors Journal*.

[10] Wisanwanichthan, T., & Thammawichai, M. (2021). A double-layered hybrid approach for network intrusion detection system using combined naive bayes and SVM. *Ieee Access*, *9*, 138432-138450.

[11] M. H. Haghighat and J. Li, Intrusion detection system using voting based neural network, Tsinghua Sci. Technol., vol. 26, no. 4, pp. 484–495, Aug. 2021.

[12] Y. Yang et al., ASTREAM: Data-stream-driven scalable anomaly detection with accuracy guarantee in IoT environment, IEEE Trans. New. Sci. Eng., early access, Mar. 8, 2022, Doi: 10.1109/TNSE.2022.3157730.

[13] F. T. Liu, K. M. Ting, and Z.-H. Zhou, Isolation-based anomaly detection, ACM Trans. Know. Discov. Data, vol. 6, no. 1, pp. 1–39, Mar. 2012.

[14] X. Zhang et al., LSHiForest: A generic framework for fast tree isolation based ensemble anomaly analysis, in Proc. IEEE 33rd Int. Conf. Data Eng. (ICDE), Apr. 2017, pp. 983–994.

[15] R. Magán-Carrión, D. Urda, I. Díaz-Cano, and B. Dorronsoro, Towards a reliable comparison and evaluation of network intrusion detection systems based on machine learning approaches, Appl. Sci., vol. 10, no. 5, p. 1775, Mar. 2020.