Peer-to-Peer Networking and Applications _####################_
https://doi.org/10.1007/s12083-025-02000-0

CAT: A simple heterogeneous ensemble learning framework for network intrusion detection

Zheng Zhang[1] · **Amit Das**[2] · **Guan Huang**[3] · **Sanjeev Baskiyar**[3]

Received: 20 July 2024 / Accepted: 28 April 2025

Abstract
Given the abundance of cyber-attacks in wireless communications, machine learning can be essential in identifying intrusions. Ensemble learning has been proven to be a well-known technique for boosting performance and reducing variance. In this paper, we propose a simple ensemble supervised machine learning system, which consists of three classifiers: one-dimensional Convolutional Neural Network, FT-Transformer, and XGBoost, to categorize network traffic as benign or malicious. To find a suitable ensemble method for these classifiers, we implemented three ensemble strategies to integrate these classifiers and assessed their performance. The experimental results demonstrate that our proposed ensemble model achieves strong performance in intrusion detection, as measured by various classification metrics. When evaluated on four benchmark datasets, the model consistently delivers high Accuracy, Precision, Recall, and F1-score. The research showcases the viability of various algorithms, the power of the self-attention model, and the significance of ensemble approaches in strengthening cyber-defense strategies and understanding modern cyber threats.

Keywords Network intrusion detection · XGBoost · Transformer · Convolutional neural network · Ensemble learning · Deep learning

1 Introduction

The explosive growth of Internet business has brought significant innovations to our daily lives. However, alongside these advancements, cybercrime has also increased due to the security vulnerabilities of servers and the complexity of network behavior. Detecting and mitigating cyber threats is a critical challenge for cybersecurity professionals. Network Intrusion Detection (NID) systems play a crucial role in identifying malicious activities and protecting network infrastructure. However, developing efficient and effective NID systems remains an ongoing challenge in both industry and academia due to the evolving nature of cyber threats and the high-dimensional, imbalanced nature of intrusion datasets.

To understand the current landscape of NID research, we conducted a bibliometric analysis using VosViewer [1] over the Web of Science database with keywords such as machine learning, deep learning, and ensemble learning. Our analysis of over 300 research papers provides insights into how frequently specific keywords appear in NID-related publications within the past five years. As illustrated in Fig. 1, the term *intrusion detection* appears in articles across multiple subject areas. The articles related to intrusion detection have strong connections with the terms *machine learning* and *deep learning*. Machine learning articles connect the term *anomaly detection* with the term *intrusion detection*, indicating that machine learning methods play an important role in cybersecurity. The term *deep learning* combines Neural Network methods such as *Convolutional Neural Networks* and *ensemble learning*, with the term *intrusion detection* indicat-

✉ Zheng Zhang
zzhang22@murraystate.edu

Amit Das
adas@una.edu

Guan Huang
gzh0040@auburn.edu

Sanjeev Baskiyar
baskisa@auburn.edu

[1] Department of Computer Science and Information Systems, Murray State University, 42071 Murray, Kentucky, USA

[2] Department of Computer Science and Information Systems, University of North Alabama, 35632 Florence, Alabama, USA

[3] Computer Science & Software Engineering, Auburn University, 36849 Auburn, Alabama, USA

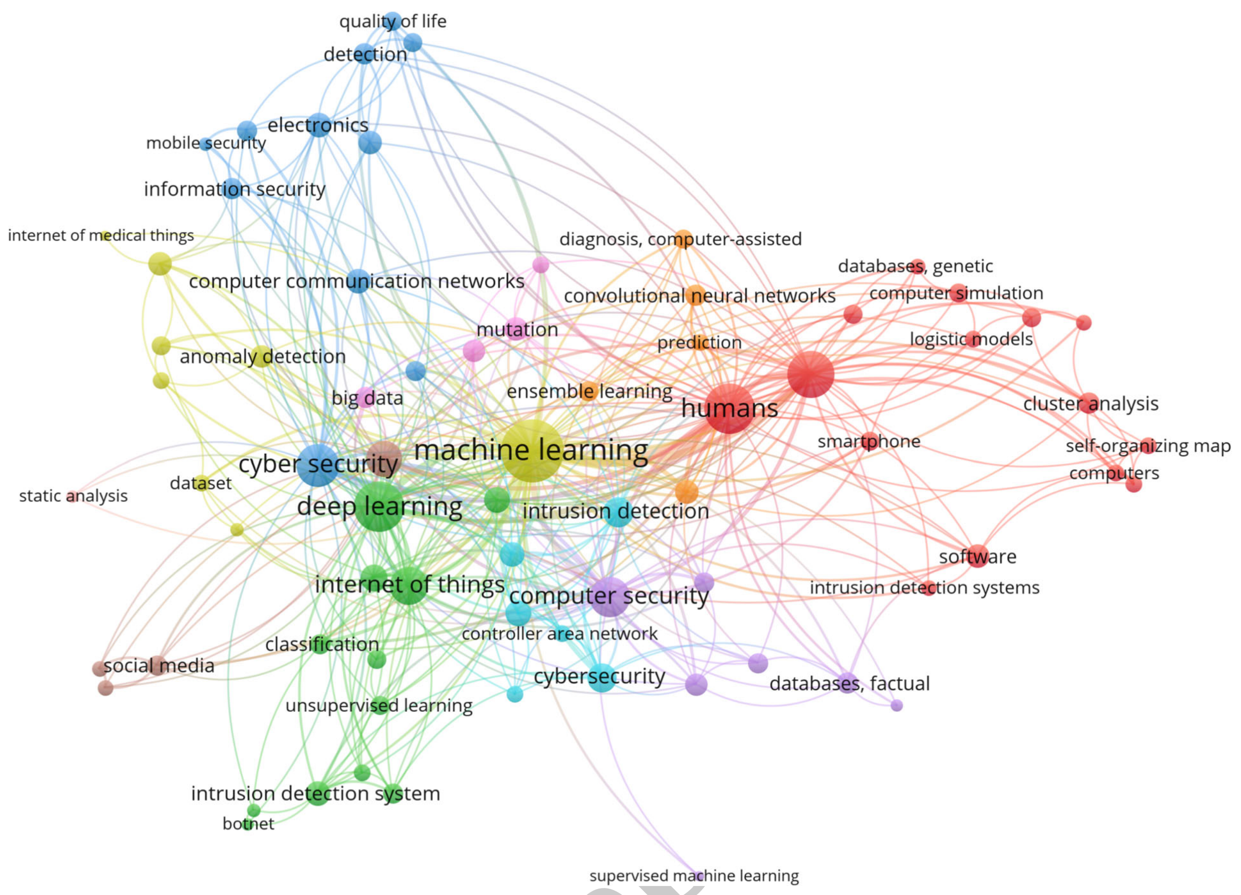

Fig. 1 Intrusion detection as transdisciplinary concept

ing a tight connection between them. Keywords, including *humans*, *smartphones*, and *quality of life*, imply a close relationship between humans and NID.

Traditional ML-based NID systems rely heavily on feature engineering to extract meaningful patterns from network data. In contrast, DL-based models, particularly those utilizing Convolutional Neural Networks (CNNs), have demonstrated the ability to automatically learn complex hierarchical features from raw network traffic [2]. Furthermore, ensemble learning techniques enhance the predictive performance of intrusion detection models by combining multiple classifiers to improve generalization and robustness. Ensemble models can be categorized into sequential approaches (e.g., XGBoost), which refine predictions iteratively, and parallel approaches (e.g., Random Forest), which leverage diverse learning models to reduce errors [3]. Recent advancements in ensemble deep learning further demonstrate the potential of combining traditional ML and DL models to achieve superior intrusion detection performance [4–6].

In this paper, we propose a new simple ensemble supervised machine learning system, combining the power of traditional ensemble and ensemble deep learning, based on one **C**onvolutional Neural Network, one **A**ttention mechanism, and one **T**ree structure, named CAT, to categorize network traffic as benign or malicious. The CAT NID system consists of three classifiers: 1-dimensional Convolutional Neural Network (1D CNN), Feature Tokenizer Transformer (FT -Transformer) [7], and XGBoost [8]. The selection of FT-Transformer, 1D CNN, and XGBoost in our ensemble framework is motivated by their complementary strengths in handling heterogeneous intrusion detection datasets. FT-Transformer is well-suited for tabular data, as it efficiently captures dependencies across categorical and numerical features using self-attention mechanisms, making it a robust alternative to traditional feature engineering-based approaches [7]. Meanwhile, 1D CNN has been demonstrated to effectively learn spatially invariant patterns from network traffic data, reducing the need for manual feature extraction and enhancing performance in classification tasks [9]. XGBoost, a powerful gradient boosting algorithm, provides strong generalization capabilities by sequentially refining predictions and mitigating overfitting through its regularization techniques [8]. By leveraging the diversity of these models, our ensemble strategy ensures a well-balanced tradeoff between feature representation learning, sequential pattern extraction, and robust decision-making. The combination of these approaches enables the model to capture both global and local patterns in intrusion datasets, leading to improved classification performance and robustness against evolving cyber threats. Experimental results from this research show that the optimized ensemble of FT-Transformer, 1D CNN, and XGBoost is superior to individual approaches and other benchmark classifers for intrusion detection in terms of classification accuracy, precision, recall, and F1-score. The contributions of this paper have been outlined by the following research outcomes:

- This study proposes a novel ensemble architecture that combines the power of traditional ensemble techniques and ensemble deep learning to categorize network traffic as benign or malicious.
- This study provides detailed investigations and analysis of various classical machine learning algorithms for network intrusion detection.
- This study utilizes the state-of-the-art self-attention model can achieve performance comparable to other models in the same domain.
- This study proves that the Tree-structured Parzen Estimator (TPE) has demonstrated its effectiveness in ensemble-based NID systems.
- This study demonstrates that an ensemble model that combines diversity models outperforms individual models creating a more robust and accurate prediction than individual models.

The remainder of this paper is organized as follows. A brief overview of relevant work on the subject of intrusion detection is provided in Section 2. Section 3 introduces the theoretical aspects of FT-Transformer, 1D CNN, and XGBoost, and the ensemble strategies of the proposed ensemble framework. The exhaustive experimental results and comparisons with previous studies using four datasets are presented in Section 4. Section 5 discusses the limitations of our approach. Finally, Section 6 concludes the paper.

2 Related works

Traditional machine learning models have played a crucial part in extracting hidden representations about the differences between normal and malicious behaviors. Liao and Vemuri [10] applied k-Nearest Neighbor (kNN) classifier that can effectively detect intrusive attacks and achieve a low false positive rate using the 1998 DARPA intrusion detection dataset [11]. Mukkamala et al. [12] demonstrated that Support Vector Machines (SVMs) and Neural Networks could deliver highly accurate results in detecting potential system intrusion using the 1998 DARPA intrusion detection dataset. Amor et al. [13] showed that even if having a simple structure, naive Bayes and decision trees provide very competitive results on the KDD'99 intrusion detection dataset [14]. Although traditional ML technologies can deliver remarkable results, they have relatively low accuracies and rely significantly on manual traffic feature design compared to Deep Learning models. Additionally, Various machine learning methods were employed in this study to detect cyber anomalies in IoT systems, with a comparative analysis revealing that neural networks outperformed other models in classification accuracy [15].

Unlike traditional ML models, DL models could eliminate feature engineering procedures and deliver high-quality learning for complex data analysis. Jiang et al. [16] used CNN to extract spatial features and bi-directional long short-term memory (BiLSTM) to extract temporal features to form a deep hierarchical network model using the NSL-KDD [14] and UNSW-NB15 datasets [17]. Wei et al. [18] proposed an optimization method for an intrusion detection classification model based on a deep belief network using the NSK-KDD dataset. A long short-term memory (LSTM)-based NID system was introduced in [19]. This method improved the performance of the intrusion detection system (IDS) using the UNSW-NB15 dataset in terms of Accuracy and F1-score. Another study [20] proposes using a deep neural network (DNN) to swiftly and accurately detect Distributed Denial of Service (DDoS) attacks in internet network traffic. Experiments on the CICDDoS2019 dataset demonstrate a 99.99% detection success rate and 94.57% accuracy in classifying attack types, indicating DNN's effectiveness against DDoS attacks. A novel anomaly-based IDS framework [21] using stacked modified Gated Recurrent Units (mGRU) is proposed in this study to detect Multi-vector DDoS attacks in mobile healthcare systems, with challenges like processing time and complexity being addressed. Additionally, A communication-efficient distributed adaptive zeroth-order optimization method, DaZoo [22], is introduced for IoT clients without gradient information, where convergence is improved, communication costs are reduced through sparsification, and state-of-the-art methods are outperformed in distributed closed-box attacks and large-scale IoT attack detection using the CICIoT2023 dataset.

Hybrid models were proven competent against malicious attacks compared to separate traditional ML and DL architectures. Vinayakumar et al. [23] proposed a highly scalable and hybrid Deep Neural Network (DNN) framework that can be used in real-time to effectively monitor the network traffic and host-level events to proactively alert possible cyberattacks. Li et al. [24] proposed a DL approach for intrusion detection using a multi-Convolutional Neural Network (multi-CNN) fusion method on the NSL-KDD dataset. Liu et al. [25] presented an adaptive network intrusion detection method using fuzzy rough set-based feature selection and A greedy algorithm-based Gaussian mixture model. Sethi et al. [26] introduced a deep reinforcement learning-based adaptive cloud IDS architecture that achieved better accuracy and less FPR compared to the state-of-the-art NID systems using the UNSW-NB15 dataset. A Graph Deep Learning framework based on Centrality measures (GDLC) [27] was proposed and tested, integrating AI techniques (CNN, LSTM, and GRU) with dynamically selected centrality measures to enhance Network Intrusion Detection Systems (NIDS), achieving up to a 7.7% improvement in detection accuracy.

While hybrid deep learning architectures improve intrusion detection, another approach that has shown promise is the use of machine learning ensemble models. Ensemble models aggregate the outputs of numerous models to reduce variance (bagging), reduce bias (boosting), or improve predictions (stacking). Ensemble methods can be divided into two groups. The first type consists of sequential ensemble approaches, such as XGBoost, which create the base learners in a sequential manner, exploiting the dependence between the base learners. Its performance can be improved by assigning higher weights to mislabeled samples. In the second type, parallel ensemble techniques are utilized where the base learners are produced concurrently, such as in Random Forest. These parallel techniques could take advantage of the differences in learning styles among the base learners since averaging significantly reduces mistakes. Ensemble techniques can also be classified into two categories: a homogenous ensemble comprises learners of the same type, while a heterogeneous ensemble is built on learners of various types [3].

Ensemble learning's ability to generalize better than single classifiers makes it an attractive solution for network intrusion detection. By combining weak classifiers, ensembles can produce results that outperform even strong individual classifiers [28]. The selection of the decision fusion function affects how well an ensemble or multiple-classifier strategy performs. The diversity of classifiers should be considered while choosing the decision function. The decision function was established based on the performance of each classifier on Accuracy, Precision, Recall, and F1-score [29]. Traditional ensemble models [29–33] have been widely used to solve NID tasks. Recently, ensemble deep learning models [4–6] have also been employed, leveraging the power of deep learning architectures to further enhance the performance and robustness of intrusion detection systems.

Several studies have validated the effectiveness of traditional ensemble methods in network intrusion detection. Das et al. [31] describe a comprehensive security solution for network intrusion detection using a traditional ensemble supervised machine learning (ML) framework and ensemble feature selection methods. Rajagopal et al. [32] present an ensemble model employing a metaclassification approach, including random forest, logistic regression, and KNN, enabled by stacked generalization. Thockchom et al. [34] propose an ensemble model constructed using lightweight machine learning models, such as Gaussian naive Bayes, logistic regression, and decision tree as base classifiers, with stochastic gradient descent as the meta-classifier. Hsu et al. [30] introduce a stacked ensemble learning-based ANIDS comprising autoencoder (AE), support vector machine (SVM), and random forest (RF) models. These studies, along with other models [29, 33, 35, 36], collectively highlight the effectiveness of traditional ensemble methods in enhancing the performance of network intrusion detection systems.

As deep learning has advanced in recent years, ensemble deep learning has emerged as a powerful approach. Lazzarini [4] proposes a novel method for intrusion detection in IoT environments using a stacking ensemble of deep learning (DL) models. Kumar [5] introduces an automated framework called DLTIF, designed to model cyber threat intelligence and identify threat types. DLTIF operates on three key schemes: a deep feature extractor, CTI-driven detection, and CTI-attack type identification. Khan's ensemble technique [6] employs a set of long short-term memory (LSTM) networks as individual learners at the first level, with a decision tree stacked on top to classify attacks and normal events. Ravi [37] presents an end-to-end model for network attack detection and classification using deep learning-based recurrent models. These studies and various models [38, 39] underscore the significant impact of deep ensemble methods in boosting the efficiency of network intrusion detection systems. Recently, A non-complex CNN1D model is introduced in [40], achieving 99.96% accuracy in detecting 15 attack classes from the Edge-IIoTset dataset and demonstrating strong generalization through k-fold cross-validation for real-world I-IoT environments. Additionally, the IoT-PRIDS model [41], which operates solely on benign network traffic, effectively identifies abnormal flows while maintaining low false alarm rates, presenting a promising solution for enhancing IoT security in practical applications.

3 Proposed system

This section proposes the detailed construction of the ensemble model. As shown in Fig. 2, the model consists of four components: input layer, multi-model layer, ensemble layer, and output layer, from left to right. In the input layer, numeric and categorical variables are preprocessed using normalization and one-hot encoding. Scaling features to a similar range is a crucial way to stabilize the gradient descent procedure during the training of Neural Networks, helping the model converge faster for a given learning rate. In the modeling layer, three decision fusion classifiers with different learning paradigms, including XGBoost, 1D CNN, and FT-Transformer, are deployed. In the ensemble layer, three different ensemble methods, including average-weighted, validation loss-weighted, and optimized-weighted ensemble (Section 3.3), are implemented to find the best ensemble

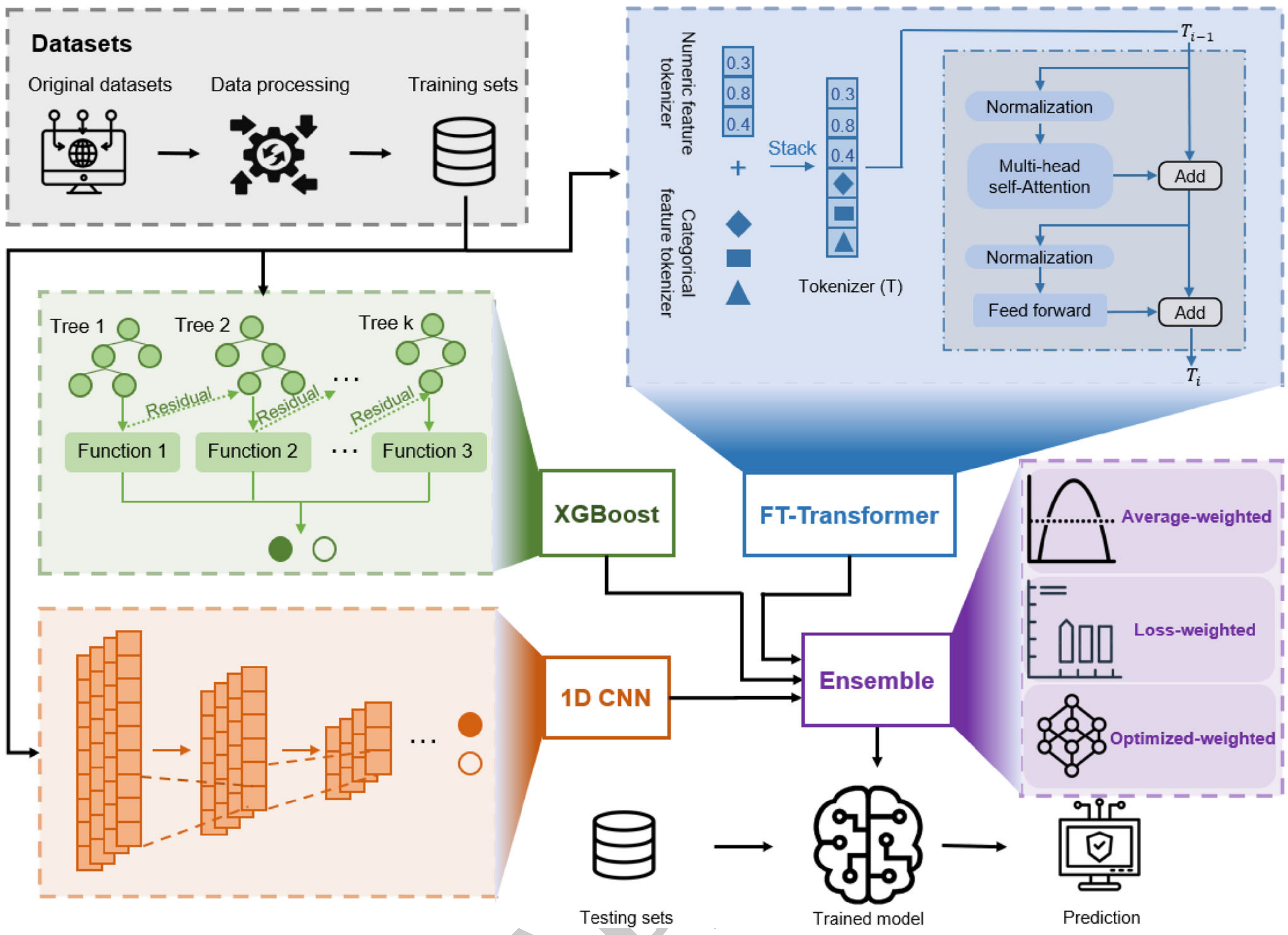

Fig. 2 Ensemble model architecture

model. The fused predictions can then be used to categorize normal networks from malicious ones.

3.1 Input layer

Data normalization is carried out at the data preprocessing stage using the one-hot encoding approach for categorical variables and the Z-score method for numeric variables.

3.1.1 One-hot encoding

Each category is converted to a vector with the values 1 and 0, indicating whether the feature is present or absent. The dimension of vectors depends on the number of categories for features. Note that the number of categories in categorical variables in all datasets is low, so one-hot encoding will not slow down the learning too much.

3.1.2 Z-Score normalization

It is performed by calculating the normalized value x_{norm} of each data sample x_i as follows:

$$x_{norm} = \frac{x_i - \mu}{\sigma}$$

where μ and σ are the mean and the standard deviation vector of the features, respectively. Normalization usually improves the numerical stability of the model and reduces training time.

3.1.3 Feature selection

Although feature selection methods are one of the fundamental concepts in machine learning, which have a significant impact on the performance of models, they suffer from seeking to identify only one solution to the problem that is insufficient for knowledge discovery. In this case, we only removed two kinds of features: those with limited relationships for classifying network flows as benign or malicious and those with too many missing values. Table 1 denotes the features we have dropped. We use all features in the NSL-KDD dataset and CICIoT2023. For the UNSW-NB15 dataset, four features including *srcip* (source IP address), *sport* (source port number), *dstip* (destination IP address), and *dsport* (destination port number) have been removed since they are only applicable to the computer infrastructure, and do not provide information that is crucial for intrusion detection. Two general-purpose features,

Table 1 Removed features

	NSL-KDD	UNSW-NB15	IDS2018	CIC2023
Irrelevant	–	srcip, sport, dstip, dsport	Timestamp	–
Too many missing values	–	ct_flw_http_mthd, is_ftp_login	–	–

ct_flw_http_mthd (number of flows that have methods) and *is_ftp_login* (whether the FTP session is accessed by user and password), have been dropped since more than half of the values are missing. For the IDS2018 dataset, the *timestamp*, which can introduce noise or irrelevant variability into data as attacks can occur at any time, has been eliminated to prevent models from making attack predictions based on time.

3.2 Modeling layer

In ensemble learning, the selection of the decision fusion function affects how well an ensemble strategy works. The anticipated level of variety among classifiers should be highly considered while determining the decision function. We deployed three decision fusion classifiers with different learning paradigms: a tree-based extreme gradient boosting algorithm, XGBoost; a one-dimensional Convolutional Neural Network, 1D CNN; and a self-attention mechanism-based model, FT-Transformer.

3.2.1 XGBoost

Decision tree-based approaches are the ideal techniques for handling tabular data due to their interpretability, reliability, and training efficiency. Decision tree algorithms direct us along a certain path in the tree based on the measurement rules. Extreme Gradient Boosting, or XGBoost, is a distributed, scalable end-to-end tree boosting technique. The primary purpose of XGBoost was to increase the effectiveness and speed of tree-based models. Compared to gradient boosting, XGBoost 1) penalizes more complicated models using both L1 and L2 regularization to avoid overfitting; 2) handles sparse features more efficiently; and 3) deploys the distributed weighted quantile sketch algorithm that finds the optimal split points among weighted datasets more effectively [8].

XGBoost [8] seeks to minimize an objective function made up of a loss function and a regularization term:

$$\begin{aligned}\text{Obj} &= \text{Loss} + \text{Regularization}\\ &= \sum_{i=1}^{n} l(y_i, \hat{y}_i) + \sum_{k=1}^{K} \Omega(f_k)\end{aligned}$$

where n is the number of instances in the training data, K is the number of trees we used, f_k is an independent tree structure, l is a differentiable convex loss function that measures how well the model fits between the prediction $\hat{y}_i$ and the target y_i, and Ω is the regularization term that measures the complexity of trees. Since:

$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$$

where $\hat{y}_i^{(t)}$ is the current model, $\hat{y}_i^{(t-1)}$ is the previous model, and f_t is the new model we are learning. We have:

$$\text{Loss} = \sum_{i=1}^{n} l(y_i, \hat{y}_i^{(t-1)} + f_t(x_i))$$

Any smooth function can be represented as an infinite sum of powers of its derivatives evaluated at a specific point using Taylor series [42]. The Taylor series of $Loss$ is:

$$\text{Loss} \simeq \sum_{i=1}^{n} [l(y_i, \hat{y}_i^{(t-1)}) + g_i f_t(x_i) + \frac{1}{2} h_i f_t^2(x_i)]$$

$g_i = \partial_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$, and $h_i = \partial^2_{\hat{y}^{(t-1)}} l(y_i, \hat{y}^{(t-1)})$ are first- and second-order gradient statistics on the loss function [8].

For the regularization term, we have:

$$\begin{aligned}\Omega(f_k) &= \gamma T + \frac{1}{2}\lambda ||w||^2\\ &= \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2\end{aligned}$$

where T is the number of leaves, and $||w||^2$ is L2 regularization of leaf scores. After removing the constant terms in $Loss$, we have:

$$\begin{aligned}Obj^{(t)} &= \sum_{i=1}^{n} [g_i f_t(\mathbf{x}_i) + \frac{1}{2} h_i f_t^2(\mathbf{x}_i)] + \gamma T + \frac{1}{2}\lambda \sum_{j=1}^{T} w_j^2\\ &= \sum_{j=1}^{T} [(\sum_{i \in I_j} g_i) w_j + \frac{1}{2}(\sum_{i \in I_j} h_i + \lambda) w_j^2] + \gamma T\\ &= \sum_{j=1}^{T} [G_j w_j + \frac{1}{2}(H_j + \lambda) w_j^2] + \gamma T\end{aligned}$$

where $G_j = \sum_{i \in I_j} g_i$, $H_j = \sum_{i \in I_j} h_i$, and I_j is the instance set of leaf j. Note that the first term in $Obj^(t)$ is the sum of T quadratic functions, for each quadratic function $G_j w_j +$

$\frac{1}{2}(H_j+\lambda)w_j^2$, we can calculate the optimal weight w_j^* of leaf j:

$$w_j^* = -\frac{G_j^2}{H_j+\lambda}$$

and the optimal value:

$$min\text{Obj}^{(t)} = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j^2}{H_j+\lambda} + \gamma T$$

We use the default setting of the XGBoost library [8] in the experiment.

3.2.2 1D CNN

A CNN is effective in finding basic patterns in data, which are then utilized to create more complicated patterns in higher layers. Over the last ten years, Convolutional Neural Networks have produced ground-breaking findings in a range of pattern recognition-related domains, including speech recognition and image processing. To identify characteristics present throughout an image, it takes advantage of the shared-weight structure of the filters, sometimes referred to as kernels. The most crucial presumption regarding issues that CNNs solve is that they should not have features that are spatially dependent [43]. This makes a CNN, especially 1D CNN, highly useful when extracting relevant features from shorter (fixed length) chunks of the entire dataset, and the feature's location within the segment is not of high relevance. The key difference among different dimensional CNNs is how the filter slides across the data.

The proposed 1D CNN model consists of 3 Convolutional layers, and the number of filters is 64, 32, and 32, respectively. Based on the experiment, the optimal kernel size would be 6×6, 3×3, and 3×3, respectively. The kernel generates a feature map by moving over the image as much as stride, which is the designated value. We set the stride to 2 to extract the feature densely. We applied batch normalization and max pooling with a pooling size of 3 for all Convolutional layers. After hyperparameter tuning, we set the learning rate to 0.01, the number of training epochs to 100 with early stopping, and chose categorical cross-entropy as the loss function and Adam as the optimizer during the training.

3.2.3 FT-transformer

Transformer [44], which was initially proposed as a sequence-to-sequence model for machine translation, is a prominent deep learning model that is frequently used in many different disciplines [45]. FT-Transformer (FT) [7], also known as the Feature Tokenizer Transformer, is a straightforward transformation of the Transformer architecture for the tabular domain. First, the Feature Tokenizer transforms the input features x to embeddings T.

$$\begin{aligned} T_j^{(num)} &= b_j^{(num)} + x_j^{(num)} \cdot W_j^{(num)} && \in \mathbb{R}^d, \\ T_j^{(cat)} &= b_j^{(cat)} + e_j^T W_j^{(cat)} && \in \mathbb{R}^d, \\ T &= \text{stack}[T^{(num)}, T^{(cat)}] && \in \mathbb{R}^{k\times d}. \end{aligned}$$

where b_j is the bias of j-th feature, $W_j^{(num)}$ and $W_j^{(cat)}$ are the vector weights for numeric and categorical features, respectively. e_j^T is a one-hot vector for the related categorical feature [7].

The model applies a stack of Transformer layers on the embeddings after transforming all features (categorical and numerical) into them. The Transformer component appends the classification ([CLS]) token [46] to T, then L Transformer layers $F_1, \ldots, F_L$ are applied.

$$T_0 = \text{stack}[[\text{CLS}], T] \quad T_i = F_i(T_{i-1})$$

PreNorm [47] is utilized for better performance and more straightforward optimization, and the final representation of the [CLS] token is used to make predictions [7]. FT-Transformer outperforms other deep learning solutions and matches the performance of Gradient Boosted Decision Trees on most tasks. We use the default setting of the FT-Transformer [7] in the experiment.

3.3 Ensemble layer

Ensemble learning is a well-known ML approach in which many models are learned and merged to address the same issue for improving performance and reducing variance. With a proper voting mechanism or weighting assignment, The combining approach has demonstrated better performance and a more resilient model when compared to a single classifier approach.

In our experiment, to construct a practical and straightforward ensemble, we suggest three linear methods: average-weighted, loss-weighted, and optimized-weighted. These methods are designed to offer a robust combination of simplicity and predictive power.

3.3.1 Average-weighted ensemble (CAT_A)

Considering the ensemble as a uniformly weighted mixture model and combining the forecasts as follows [48]:

$$p(y|x) = \sum_{k=1}^{K} p_{\theta_m}(y|x,\theta_m)$$

Algorithm 1 CAT Algorithm: Ensemble Learning with TPE Optimization.

```
Input  : Set of base learners $\mathcal{B} = \{B_1, B_2, ..., B_n\}$ where
         $B_i \in$ {XGBoost, FT-Transformer, 1D CNN};
         Training dataset: $\mathcal{D}_{\text{train}} = \{(x_j, y_j)\}_{j=1}^{N_{\text{train}}}$;
         Ensemble training dataset:
         $\mathcal{D}_{\text{ensemble}} = \{(x_j, y_j)\}_{j=1}^{N_{\text{ensemble}}}$;
         Testing dataset: $\mathcal{D}_{\text{test}} = \{x_j\}_{j=1}^{N_{\text{test}}}$;
Output: Test predictions: $\hat{Y}_{\text{test}} = \{\hat{y}_j\}_{j=1}^{N_{\text{test}}}$;
1  Phase 1: Base Learner Training
2  for each base learner $B_i \in \mathcal{B}$ do
3      $B_i \leftarrow \arg\min_{\theta_i} \mathcal{L}_{B_i}(\mathcal{D}_{\text{train}}, \theta_i)$, where $\mathcal{L}_{B_i}$ is the loss
       function specific to $B_i$;
4      Store trained model parameters $\theta_i^*$ for $B_i$;
5  end for
6  Phase 2: Ensemble Model Training
7  Define ensemble model $\mathcal{E}(\cdot; \Theta_{\mathcal{E}})$ parameterized by $\Theta_{\mathcal{E}}$;
8  Train $\mathcal{E}$ using $\mathcal{D}_{\text{ensemble}}$ by solving:
   $\Theta_{\mathcal{E}}^* = \arg\min_{\Theta_{\mathcal{E}}} \mathbb{E}_{(x_j,y_j)\sim\mathcal{D}_{\text{ensemble}}} \mathcal{L}_{\mathcal{E}}(y_j, \mathcal{E}(\{B_i(x_j)\}_{i=1}^n; \Theta_{\mathcal{E}}))$
   where $\mathcal{L}_{\mathcal{E}}$ is the loss function for the ensemble;
9  Optimize $\Theta_{\mathcal{E}}$ using Tree-structured Parzen Estimator (TPE);
10 Store optimized ensemble parameters $\Theta_{\mathcal{E}}^*$;
11 Phase 3: Prediction on Test Data
12 for each test instance $x_j \in \mathcal{D}_{test}$ do
13     Compute base model predictions:
       $\mathcal{P}_j = \{P_i(x_j) \mid P_i = B_i(x_j), \forall B_i \in \mathcal{B}\}$
       Compute final ensemble prediction using aggregation
       function $\mathcal{A}(\mathcal{P}_j)$:
       $\hat{y}_j = \mathcal{E}(\mathcal{P}_j; \Theta_{\mathcal{E}}^*)$
       Store $\hat{y}_j$ in $\hat{Y}_{\text{test}}$;
14 end for
15 return $\hat{Y}_{test}$
```

where θ_m is the hyper-parameter for the model and K=3 since we have three individual models.

3.3.2 Loss-weighted ensemble (CAT_L)

A weighted average of the predictions from each trained model. Instead of the normalized validation loss [48], we use the normalized negative logarithmic validation loss of each model serves as its relative weight.

$$p(y|x) = \sum_{k=1}^{K} l_k p_{\theta_m}(y|x, \theta_m)$$

where $l_k = -log(l_k^{val})$ and l_k^{val} is the validation loss for the k-th model.

3.3.3 Optimized-weighted ensemble (CAT_O)

A weighted average of the predictions from each trained model.

$$p(y|x) = \sum_{k=1}^{K} w_k p_{\theta_m}(y|x, \theta_m)$$

where w_k is the weight for the kth model found by Tree-Structured Parzen Estimator (TPE) [49]. The TPE algorithm is created to optimize the quantization hyperparameters to identify the best feasible quantization configuration that improves latency while achieving the intended accuracy objective. To recommend the next set of hyperparameters to be assessed, TPE is an iterative procedure that builds a probabilistic model using the history of evaluated hyperparameters.

The proposed intrusion detection system can now be defined in Algorithm 1. The dataset is split into training, ensembling, and testing sets with a 60/20/20 split. First, each base learner is trained on the provided training data to learn underlying patterns and characteristics of the network traffic. The trained base learners are then stored as members of the ensemble. Next, the ensemble model is trained on the ensembling data using TPE for hyperparameter optimization. Once the ensemble model is trained, it proceeds to make predictions on the testing data. For each instance in the testing data, predictions (probabilities) are generated using each member of the ensemble. These predictions are then aggregated, either through averaging, log loss-weighted aggregation, or using optimal weights determined by TPE during training. The aggregated prediction represents the ensemble's collective decision on the class label for the instance. This process is repeated for all instances in the testing data, and the final test predictions are obtained. By combining the strengths of multiple base learners through ensemble methods, the CAT algorithm proves to be a powerful tool for effective network intrusion detection, providing enhanced security in complex and dynamic network environments.

4 Experiments

In this section, we use Python to implement the software prototype, combining three classifiers, XGBoost, 1D CNN, and FT-Transformer, with three ensemble strategies, CAT_A, CAT_L, and CAT_O. We conduct all evaluations on a Linux server equipped with an NVIDIA H100 GPU and validate the performance of the proposed architecture through verification and comparative experiments conducted on four datasets: NSL-KDD, UNSW-NB15, CIC-IDS2018 and CICIoT2023.

Table 2 NSL-KDD dataset

		KDDTrain$^+$	KDDTest$^+$	No. Samples/ Distribution (%)
Normal		67,343	9,711	77,054/ 51.88
Attacks	DoS	45,927	7,458	53,385/ 35.95
	Probe	11,656	2,421	14,077/ 9.48
	R2L	995	2,754	3,749/ 2.52
	U2R	52	200	252/ 0.17
Total		125,973	22,544	148,517

4.1 Dataset

Benchmark-labeled datasets serve as a solid foundation for assessing and contrasting the quality of various NID systems. In this section, we use NSL-KDD, UNSW-NB15, CIC-IDS2018 and CICIoT2023 datasets, which are widely used in the literature, to test the performance of the proposed ensemble architecture.

NSL-KDD This dataset [50] is the revised and refined version of the KDD Cup'99 dataset [51], one of the most popular and widely used datasets for IDS, by removing duplicates and creating more sophisticated subsets. The dataset was gathered with the help of the Canadian Institute for Cybersecurity and the University of New Brunswick. Each sample in NSK-KDD contains 41 features and is labeled as either normal or attack. The attacks are classified into four different types: Denial of Service (DoS), Probe, Remote to Local (R2L), and User to Root (U2R).

UNSW-NB15 The UNSW-NB15 dataset, created by the Australian Center for Cybersecurity, contains both normal and malicious network traffic in packet-based format [17]. The features of the UNSW-NB15 data collection are produced using established and new methodologies. Each sample contains 49 features, and the dataset has nine attack types, including Generic, Exploits, Fuzzers, DoS, Reconnaissance, Analysis, Backdoors, Shellcode, and Worms.

CSE-CIC-IDS2018 (IDS2018) This dataset was jointly created by the Communications Security Establishment (CSE) and the Canadian Institute for Cybersecurity (CIC) in 2018 and is the most recent publicly available dataset with big data. Sharafaldin et al. systematically build datasets using the concept of profiles, and these datasets will include explicit descriptions of incursions as well as abstract distribution models for applications, protocols, or lower-level network entities. The final dataset consists of 80 features and six different attack scenarios [52].

CICIoT2023 This dataset is a real-time dataset for IoT security research, designed to support the development of intrusion detection systems. It consists of network traffic data collected from 105 IoT devices in an extensive topology, simulating real-world environments. The dataset includes 33 different attacks categorized into seven types: DDoS, DoS, reconnaissance, web-based, brute force, spoofing, and Mirai-based attacks. Attack data was generated by malicious IoT devices targeting other IoT devices, making it unique in its focus [53].

Table 2 through Table 5 present comprehensive information on sample size, attack scenarios, and distribution of the datasets. Notably, the CICIoT2023, IDS2018 and the UNSW-NB15 datasets contain a greater number of samples and attack scenarios than the NSL-KDD dataset. In terms of balance, only NSL-KDD is evenly distributed, while UNSW-NB15 has a high-class imbalance which may lead to low accuracy and an increased false positive rate in the system. Similarly, the CICIoT2023 dataset exhibits a significant class imbalance, with attack samples overwhelmingly outnumbering normal traffic, particularly dominated by DDoS attacks, which account for over 72% of the dataset.

4.2 Evaluation Metrics

This section introduces Accuracy, Precision, Recall, and F1-score to measure the performance of the NID system. As the fundamental component for each measurement, True Positive (TP) is the correct classification of the normal flow, False Positive (FP) represents the incorrect classification of an intruder to a normal flow, True Negative (TN) indicates an intruder classified correctly, and False Negative (FN) denotes a sample where a normal flow is incorrectly classified as an intruder.

Table 3 UNSW-NB15 dataset

		No. Samples	Distribution (%)
Normal		2,218,761	87.351
Attacks	Generic	215,481	8.483
	Exploits	44,525	1.753
	Fuzzers	24,246	0.955
	DoS	16,353	0.644
	Reconnaissance	13,987	0.551
	Analysis	2,677	0.105
	Backdoors	2,329	0.092
	Shellcode	1,511	0.059
	Worms	174	0.007
Total		2540044	

Table 4 CIC-IDS2018 dataset

		No. Samples	Distribution (%)
Normal		2,856,035	62.535
Attacks	DoS	1,289,544	28.235
	Bot	286,191	6.266
	Infiltration	93,063	2.038
	DDoS	41,730	0.914
	Brute force	513	0.011
	SQL injection	53	0.001
Total		4,567,129	

Accuracy indicates the proportion of correctly classified samples to the total number of samples. Accuracy is the most common measurement for model evaluation, especially for a balanced dataset.

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

Precision is the classifier's intuitive ability to not label a negative sample as positive. Precision is an excellent measure to determine when the costs of FP are high. The evaluation metrics are defined as follows:

$$Precision = \frac{TP}{TP+FP}.$$

Recall is used to assess how well a model performs in terms of accurately counting true positives among all real positive values. Recall is a good measure to determine when the costs of FN are high. Note that the Precision and Recall scores are helpful indicators of the success of prediction when the classes are very imbalanced.

$$Recall = \frac{TP}{TP+FN}.$$

F1-score which is also called Harmonic Precision-Recall Mean, depicts a balance between Precision and Recall. F1-score is a relevant statistic in addition to accuracy when a class distribution is uneven (a large number of actual negatives).

Table 5 CICIoT2023 dataset

		No. Samples	Distribution (%)
Normal		1,098,195	2.352
Attacks	DDoS	33,984,560	72.793
	Dos	8,090,738	17.330
	Mirai	2,634,124	5.642
	Spoofing	486,504	1.042
	Recon	354,565	0.759
	Web	24,829	0.053
	BruteForce	13,064	0.028
Total		46,686,579	

Table 6 Experimental results of CAT for binary classification on NSL-KDD dataset

	XGBoost	1D CNN	FT	CAT_A	CAT_L	CAT_O
Accuracy (%)	99.52	98.64	97.32	99.43	99.46	99.68
Precision (%)	99.53	98.65	97.35	99.62	99.64	99.67
Recall (%)	99.53	98.69	97.33	99.40	99.39	99.68
F1-score (%)	99.52	98.68	97.33	99.41	99.45	99.67

$$F1-score = \frac{2}{\frac{1}{Precision}+\frac{1}{Recall}} = \frac{TP+TN}{TP+FP+FN+TN}$$

4.3 Performance analysis of ensemble model

Experiments have been designed to study the performance of the ensemble model for binary and multi-class classification over four datasets. Note that each number reported in Table 6 through Table 13 is the mean from 20 trials with different seeds. The confusion matrix generated by the model on the four datasets is shown in Figs. 3, 4, 5 and 6, respectively. The experimental results show that most samples are concentrated on the diagonal of the confusion matrix, indicating that the overall classification performance is very high. After careful fine-tuning, the comparison of Accuracy, Precision, Recall, and F1-score for both binary and multiclass classification of the three classifiers, the average, the loss, and the optimized-weighted ensemble model across the four datasets are shown in Tables 6, 7, 8, 9, 10, 11, 12 and 13, respectively. All four measurements for each classifier are high. The XGBoost

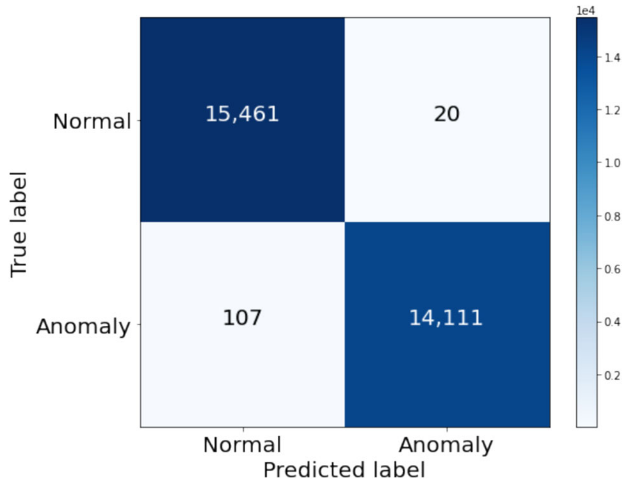

Fig. 3 Confusion matrix of CAT_O on NSL-KDD dataset

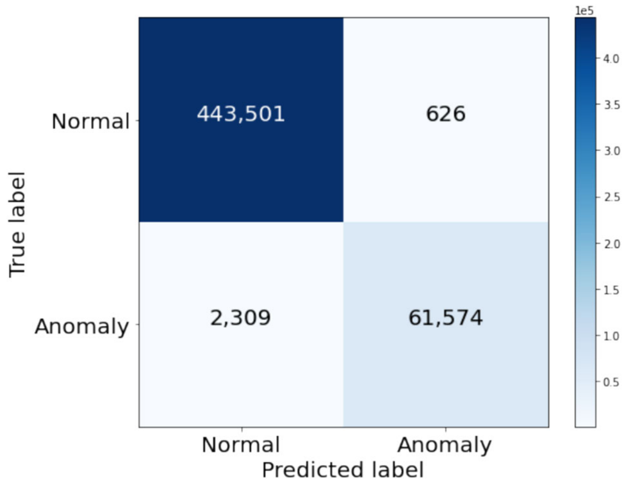

Fig. 4 Confusion matrix of CAT_O on UNSW-NB15 dataset

classifier outperforms 1-D CNN and FT-Transformer. We did not search for an optimal solution when combining individual classifiers in CAT_A and CAT_L, as the performances of 1-D CNN and FT-Transformer are relatively low. The ensemble models should have better performance than each learner however we observe that the CAT_A and CAT_L models do not show significant improvement over each classifier, and especially over XGBoost. We found an optimal solution when combining each base learner in CAT_O. CAT_O performs the best because it employs the TPE method to find the optimal weight combinations for each base learner, ensuring the most effective integration of their strengths. Meanwhile, CAT_O's fine-tuning allows it to leverage the unique contributions of XGBoost, 1-D CNN, and FT-Transformer in a balanced manner. This tailored approach maximizes the ensemble's performance across all four measurements on all four datasets, resulting in superior classification outcomes.

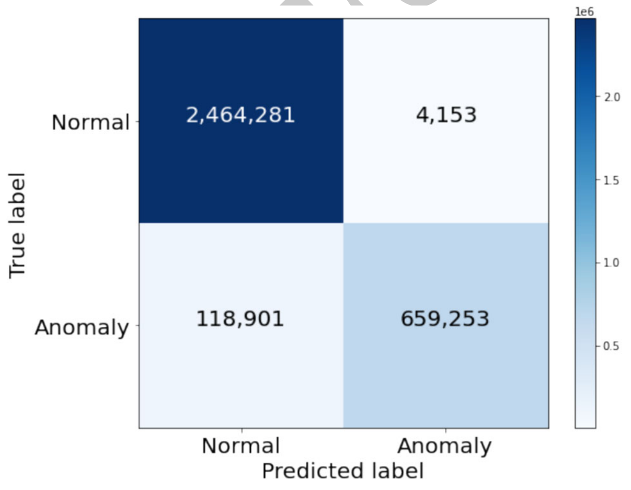

Fig. 5 Confusion matrix of CAT_O on IDS2018 dataset

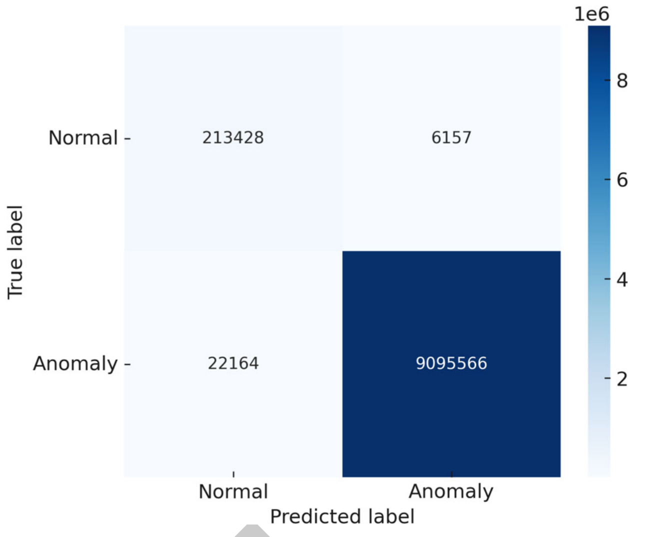

Fig. 6 Confusion matrix of CAT_O on CICIoT2023 dataset

4.4 Comparison

In this section, to demonstrate the effectiveness of the CAT_O model, our best ensemble model, we compare it with 1) other supervised machine learning models, 2) other hybrid network intrusion detection models, and 3) another ensemble model that has three tree-based classifiers: random forest, decision tree, and XGBoost.

4.4.1 CAT_O vs. ML models

Traditional supervised machine learning algorithms have been widely used for NID systems. We compare the CAT_O with several most commonly used ML models: Decision Tree (DT), Random Forest (RF), Adaboost (Ada), Naive Baye (NB), K-Nearest Neighbor (KNN), and Multi-layer Perceptron (MLP).

- **Decision Tree classifier**: It is a Supervised Machine Learning Algorithm that works by recursively partitioning the input data into subsets based on the values of its features. The algorithm selects the feature that best separates the data into the classes it is trying to predict based on a metric such as information gain or Gini impurity at each node of the tree. This process continues until a final decision is reached at a leaf node, which represents a prediction about which class the input data belongs to. Decision tree classifiers are interpretable, capable of handling numerical and categorical data, and can be used for binary and multi-class classification tasks.

Table 7 Experimental results of CAT for binary classification on UNSW-NB15 dataset

	XGBoost	1D CNN	FT	CAT_A	CAT_L	CAT_O
Accuracy (%)	99.34	99.10	99.07	99.37	99.37	99.43
Precision (%)	99.36	99.09	99.09	99.36	99.36	99.42
Recall (%)	99.34	99.10	99.05	99.37	99.38	99.42
F1-score (%)	99.35	99.08	99.07	99.56	99.37	99.42

Table 8 Experimental results of CAT for binary classification on IDS2018 dataset

	XGBoost	1D CNN	FT	CAT_A	CAT_L	CAT_O
Accuracy (%)	96.17	95.60	94.15	96.16	96.20	96.34
Precision (%)	96.32	95.97	94.50	96.23	96.27	96.47
Recall (%)	95.97	95.83	93.82	96.16	96.20	96.32
F1-score (%)	96.10	95.93	94.05	96.17	96.21	96.41

Table 9 Experimental results of CAT for binary classification on CICIoT2023 dataset

	XGBoost	1D CNN	FT	CAT_A	CAT_L	CAT_O
Accuracy (%)	99.67	99.22	99.16	99.53	99.59	99.70
Precision (%)	98.19	94.58	91.91	99.56	99.61	99.71
Recall (%)	95.07	89.64	90.22	99.53	99.59	99.70
F1-score (%)	96.58	91.96	91.05	99.54	99.60	99.70

Table 10 Experimental results of CAT for multi-class classification on NSL-KDD dataset

Type	Metrics	XGBoost	1D CNN	FT	CAT_A	CAT_L	CAT_O
Normal	Accuracy	99.66	99.24	97.18	99.33	99.46	99.33
	Precision	99.66	98.22	98.40	99.33	99.46	99.66
	Recall	99.70	99.24	97.18	99.53	99.63	99.71
	F1-score	99.68	98.73	97.79	99.43	99.54	99.69
DoS	Accuracy	99.87	99.40	99.26	99.82	99.84	99.87
	Precision	99.87	99.81	99.34	99.82	99.84	99.87
	Recall	99.93	99.40	99.26	99.92	99.92	99.94
	F1-score	99.90	99.61	99.30	99.87	99.88	99.91
Probe	Accuracy	99.50	97.62	97.33	98.89	99.17	98.89
	Precision	99.50	98.78	90.64	99.10	99.17	99.57
	Recall	99.82	97.62	97.33	98.89	99.46	99.82
	F1-score	99.66	98.19	93.87	98.99	99.32	99.69
R2L	Accuracy	97.18	82.61	79.38	95.34	96.19	97.34
	Precision	97.18	89.88	80.95	95.34	96.19	97.18
	Recall	95.14	82.61	79.38	91.33	92.77	95.41
	F1-score	96.15	86.09	80.16	93.29	94.45	96.29
U2R	Accuracy	76.92	69.55	42.86	75.00	80.02	84.27
	Precision	76.92	61.94	49.68	75.00	80.02	84.27
	Recall	64.52	69.55	42.86	38.71	38.71	61.29
	F1-score	70.18	65.52	46.02	51.06	52.18	70.97

Table 11 Experimental results of CAT for multi-class classification on UNSW-NB15 dataset

Type	Metrics	XGBoost	1D CNN	FT	CAT_A	CAT_L	CAT_O
Normal	Accuracy	100.00	100.00	100.00	100.00	100.00	100.00
	Precision	100.00	99.99	100.00	100.00	100.00	100.00
	Recall	100.00	100.00	100.00	100.00	100.00	100.00
	F1-score	100.00	99.99	100.00	100.00	100.00	100.00
Generic	Accuracy	99.22	98.68	99.78	99.49	99.27	99.53
	Precision	97.96	97.01	95.87	99.49	99.27	99.53
	Recall	99.22	98.68	99.78	97.30	97.64	98.93
	F1-score	98.59	97.84	97.78	98.38	98.45	99.23
Exploits	Accuracy	69.21	80.05	67.03	72.83	73.93	73.54
	Precision	74.61	65.91	70.86	72.83	73.93	73.54
	Recall	69.21	80.05	67.03	72.96	72.24	72.91
	F1-score	71.81	72.30	68.89	72.89	73.08	73.22
Fuzzers	Accuracy	76.28	63.25	65.32	70.37	69.92	77.05
	Precision	82.06	92.35	78.15	70.37	69.92	77.05
	Recall	76.28	63.25	65.32	87.84	88.35	89.84
	F1-score	79.07	75.08	71.16	78.14	78.06	82.95
DoS	Accuracy	44.94	43.77	44.99	45.70	46.06	48.28
	Precision	49.49	53.80	58.10	45.70	46.06	58.54
	Recall	44.94	43.77	44.99	52.41	53.29	52.23
	F1-score	47.11	48.27	50.71	48.82	49.41	55.21
Reconnaissance	Accuracy	92.50	82.73	54.28	85.44	87.71	90.61
	Precision	82.41	83.58	74.71	85.44	87.71	90.61
	Recall	92.50	82.73	54.28	82.70	82.99	82.70
	F1-score	87.16	83.15	62.87	84.05	85.29	86.47
Analysis	Accuracy	71.28	84.62	69.23	93.21	93.05	93.67
	Precision	69.01	67.59	63.32	93.21	93.05	93.67
	Recall	61.28	84.62	68.23	62.45	69.57	69.90
	F1-score	64.92	75.15	68.73	74.79	79.61	80.06
Backdoors	Accuracy	65.79	62.00	60.98	66.13	65.03	67.26
	Precision	69.01	68.56	62.32	67.34	68.54	68.92
	Recall	62.28	62.00	64.43	62.61	65.02	65.65
	F1-score	65.47	65.12	63.36	64.89	66.73	67.25
Shellcode	Accuracy	62.07	53.23	56.32	68.75	59.62	69.29
	Precision	48.00	44.00	48.31	68.75	59.62	69.29
	Recall	62.07	53.23	56.32	59.33	41.33	59.67
	F1-score	54.14	48.18	52.01	63.69	48.82	64.12
Worms	Accuracy	85.01	77.23	70.82	83.25	82.22	86.16
	Precision	81.81	72.12	68.58	83.25	82.22	86.16
	Recall	85.01	77.23	70.82	81.71	81.19	82.45
	F1-score	83.38	74.59	69.68	82.47	81.70	84.26

Table 12 Experimental results of CAT for multi-class classification on IDS2018 dataset

Type	Metrics	XGBoost	1D CNN	FT	CAT_A	CAT_L	CAT_O
Normal	Accuracy	95.92	94.82	95.58	95.25	95.46	95.97
	Precision	95.92	99.95	99.20	95.25	95.46	95.97
	Recall	99.44	94.82	95.58	99.90	99.78	99.97
	F1-score	97.65	97.32	97.36	97.52	97.57	97.93
DoS	Accuracy	96.39	96.47	94.78	96.30	96.37	96.59
	Precision	96.39	87.35	99.16	96.30	96.39	96.49
	Recall	89.46	96.47	85.01	89.46	89.48	89.49
	F1-score	92.80	91.68	91.54	92.76	92.81	92.86
Bot	Accuracy	100.00	99.80	97.77	100.00	100.00	100.00
	Precision	99.97	95.16	98.05	100.00	99.98	100.00
	Recall	99.63	99.80	97.77	99.63	99.63	99.63
	F1-score	99.81	97.43	97.91	99.81	99.81	99.81
Infiltration	Accuracy	61.29	73.19	56.30	67.50	63.58	71.57
	Precision	61.29	70.21	53.01	67.50	63.58	71.57
	Recall	67.22	65.99	56.30	73.50	71.21	69.81
	F1-score	64.12	68.03	54.61	70.37	67.18	70.68
DDoS	Accuracy	100.00	99.93	98.81	99.96	99.96	100.00
	Precision	100.00	98.18	99.12	99.96	99.96	100.00
	Recall	100.00	99.93	98.81	99.37	100.00	100.00
	F1-score	100.00	99.04	98.97	99.67	99.98	100.00
Bruteforce	Accuracy	84.38	82.12	96.72	84.23	84.22	84.38
	Precision	84.38	95.18	71.95	84.23	84.25	84.58
	Recall	94.46	82.12	96.72	94.46	94.43	94.49
	F1-score	89.13	88.17	82.51	89.05	89.02	89.26
SQL Injection	Accuracy	94.32	67.73	65.43	68.81	94.13	94.81
	Precision	92.91	72.15	68.81	68.87	93.12	93.23
	Recall	93.33	67.72	65.43	68.81	93.22	93.21
	F1-score	93.12	69.86	67.08	68.84	93.17	93.22

Table 13 Experimental results of CAT for multi-class classification on CICIoT2023 dataset

Type	Metrics	XGBoost	1D CNN	FT	CAT_A	CAT_L	CAT_O
Normal	Accuracy	91.09	78.24	69.51	89.27	91.01	91.11
	Precision	95.63	90.82	98.16	89.26	91.03	90.80
	Recall	91.09	78.24	69.15	96.59	96.50	96.69
	F1-score	93.30	84.06	81.14	92.78	93.68	93.90
Brute Force	Accuracy	100.00	100.00	100.00	100.00	100.00	100.00
	Precision	66.67	75.00	75.02	100.00	100.00	100.00
	Recall	100.00	100.00	100.00	75.00	66.67	66.67
	F1-score	80.02	85.71	85.73	85.71	80.00	86.00
DDoS	Accuracy	99.99	99.44	86.65	99.95	99.98	99.99
	Precision	99.99	99.59	94.25	99.96	99.97	99.99
	Recall	99.99	99.44	86.65	99.96	99.99	99.99
	F1-score	99.99	99.52	90.29	99.96	99.99	99.99

Table 13 continued

Type	Metrics	XGBoost	1D CNN	FT	CAT_A	CAT_L	CAT_O
DoS	Accuracy	99.96	98.51	63.44	99.87	99.56	99.96
	Precision	99.95	97.82	40.34	99.89	99.96	99.96
	Recall	99.96	98.51	63.44	99.90	99.95	99.95
	F1-score	99.96	98.16	49.32	99.89	99.96	99.96
Mirai	Accuracy	100.00	99.37	99.96	100.00	100.00	100.00
	Precision	100.00	99.25	99.17	100.00	100.00	100.00
	Recall	100.00	99.37	99.96	99.84	100.00	100.00
	F1-score	100.00	99.31	99.56	99.92	100.00	100.00
Recon	Accuracy	90.82	58.78	74.79	85.84	88.97	91.68
	Precision	80.85	70.70	49.30	85.83	88.99	89.57
	Recall	90.82	58.78	74.79	81.97	81.97	82.25
	F1-score	85.54	64.19	59.42	83.86	85.34	85.76
Spoofing	Accuracy	87.17	83.40	82.02	90.05	90.24	90.89
	Precision	85.65	44.78	31.74	90.07	90.23	90.44
	Recall	87.17	83.40	82.02	80.65	84.35	88.34
	F1-score	86.40	58.27	45.77	85.09	87.19	89.38
Web-based	Accuracy	94.12	95.23	93.85	100.00	100.00	100.00
	Precision	69.57	72.81	68.92	100.00	100.00	100.00
	Recall	94.12	92.45	95.30	81.74	65.22	60.87
	F1-score	80.04	81.41	80.21	89.95	78.95	75.68

Table 14 Model comparison between single ML model and CAT_O on NSL-KDD dataset

	DT	RF	Ada	NB	KNN	MLP	CAT_O
Accuracy (%)	99.36	99.51	96.83	88.02	99.23	98.79	99.68
Precision (%)	99.36	99.52	96.84	88.04	99.24	98.79	99.67
Recall (%)	99.37	99.52	96.83	88.03	99.24	98.78	99.68
F1-score (%)	99.37	99.52	96.83	88.03	99.24	98.79	99.67

Table 15 Model comparison between single ML model and CAT_O on UNSW-NB15 dataset

	DT	RF	Ada	NB	KNN	MLP	CAT_O
Accuracy (%)	99.34	99.30	99.13	90.08	99.06	99.15	99.43
Precision (%)	99.33	99.31	99.12	91.07	99.05	99.16	99.42
Recall (%)	99.34	99.30	99.12	90.08	99.05	99.15	99.42
F1-score (%)	99.33	99.30	99.12	89.01	99.05	99.15	99.42

Table 16 Model comparison between single ML model and CAT_O on IDS2018 dataset

	DT	RF	Ada	NB	KNN	MLP	CAT_O
Accuracy (%)	93.30	94.20	90.91	61.53	93.21	92.35	96.34
Precision (%)	93.31	94.34	91.16	89.86	93.25	93.02	96.47
Recall (%)	93.31	94.33	90.91	61.54	93.21	92.35	96.32
F1-score (%)	93.31	94.19	90.93	66.30	93.22	92.37	96.41

Table 17 Model comparison between single ML model and CAT_O on CICIoT2023 dataset

	DT	RF	Ada	NB	KNN	MLP	CAT_O
Accuracy (%)	99.59	99.67	99.58	76.57	99.00	99.18	99.70
Precision (%)	99.61	99.68	99.59	61.71	99.04	99.21	99.71
Recall (%)	99.61	99.67	99.59	49.33	99.01	99.18	99.70
F1-score (%)	99.61	99.68	99.59	61.71	99.04	99.21	99.70

- **Adaboost classifier**: It performs by combining several "weak" learners into a "strong" learner. Each weak learner is trained on a subset of the training data, and AdaBoost assigns weights to the misclassified examples to increase their importance in the next round of training. In this way, the algorithm could focus on the samples that are difficult to classify and attempt to improve the overall performance of the model.
- **Naive Bayes classifier**: The Naive Bayes classifier is a probabilistic machine learning algorithm that is widely used due to its simplicity, speed, and efficiency. It is based on Bayes' theorem, which calculates the probability of a hypothesis based on the probability of the data and prior knowledge. The name "Naive" refers to the algorithm's presumption that the characteristics are conditionally independent given the class label. The classifier employs Bayes' theorem to determine the likelihood of each class given a new instance's feature values after learning the conditional probability distribution of each feature given the class label from the training data.
- **K-Nearest Neighbor classifier**: KNN is a simple, effective algorithm that can handle complex decision boundaries. The classifier locates the k examples that are most similar to the new instance being classed in the training data, and it then determines the class label based on the majority of these closest neighbors. The technique uses a distance metric, such as Euclidean distance, to calculate the separation between instances. However, it can be sensitive to the choice of the distance metric, the number of neighbors k, and the scale of the features.

The results in Tables 14, 15, 16 and 17 show that all classifiers we tested performed well in categorizing network traffic as benign or malicious. CAT_O outperforms all the machine learning models on four datasets for all measurements.

- **Random forest** Random Forest works by combining a large number of decision trees, each of which is trained on a random subset of the input data and input features, then aggregating the predictions of these trees to improve classification performance. Random forest classifiers could handle high-dimensional data, non-linear relationships, and noisy data. It is also less prone to overfitting than individual decision trees.
- **Multi-layer perceptron** A multi-layer perceptron (MLP) classifier consists of multiple layers of interconnected nodes, including an input layer, an output layer, and one or more hidden layers. Each node applies an activation function to the weighted sum of its inputs, and the output of each layer serves as the input for the next layer. The weights of the MLP are learned through backpropagation, which adjusts the weights to minimize the error between the predicted and true outputs. MLP could handle huge, complicated datasets and learn non-linear correlations.

4.4.2 CAT_O vs. Tree-based ensemble

Ensemble diversity, the difference among the individual classifiers, is a fundamental issue in ensemble methods [54]. Some level of diversity in predictions is desired to build an effective ensemble. To demonstrate the importance of the diversity of classifiers in ensemble models, we build a tree-based ensemble model called Ensemble_T, which consists of one Decision tree, one Random Forest, and one XGBoost. As shown in Table 18, although the RF and DT model in Ensemble_T outperforms 1D CNN and FT-Transformer in CAT_O as a single classifier, the performances of CAT_O are better than those of Ensemble_T on four datasets. This experiment proved that the diversity of classifiers is important when choosing the decision function.

Table 18 Comparison between Tree-based ensemble (ET) and CAT_O

	NSL-KDD		UNSW-NB15		IDS2018		CICIoT2023	
	ET	CAT_O	ET	CAT_O	ET	CAT_O	ET	CAT_O
Accuracy	99.59	99.68	99.33	99.43	94.88	96.34	99.62	99.70
Precision	99.58	99.67	99.32	99.42	95.01	96.47	99.63	99.71
Recall	99.59	99.68	99.33	99.42	94.88	96.32	99.63	99.70
F1-score	99.58	99.67	99.33	99.42	94.87	96.41	99.63	99.70

5 Limitations

Despite the strong performance of the proposed CAT ensemble model for network intrusion detection, there are several limitations that should be acknowledged:

Computational Complexity The combination of three diverse classifiers (XGBoost, 1D CNN, and FT-Transformer) requires significant computational resources for both training and inference. This may limit the feasibility of deploying CAT in real-time, resource-constrained environments. XGBoost is known for its efficiency, with a time complexity of $O(K \cdot d \cdot n \log n)$, where K is the number of boosting rounds, d is the depth of the trees, and n is the number of samples. In contrast, Transformer-based models like FT-Transformer have a time complexity of $O(n^2 \cdot d)$ due to the self-attention mechanism, making them significantly more computationally expensive, especially for large datasets.

Adaptability to Evolving Threats Cyber threats constantly evolve, with attackers developing novel intrusion techniques. CAT relies on historical attack patterns for detection, meaning that zero-day attacks or new variants may not be effectively detected without continuous model updates.

Alternative Ensemble Methods While the CAT model utilizes a heterogeneous ensemble approach, other ensemble strategies, such as stacking with meta-learners, could be explored to improve performance and efficiency. Experimenting with different ensemble architectures may provide better generalization and computational trade-offs.

6 Conclusion

In this paper, we proposed CAT, a simple ensemble supervised machine learning system that consists of FT-Transformer, 1D CNN, and XGBoost, to categorize network traffic as benign or malicious. We deployed three ensemble strategies, and the experiment results show that the CAT_O model offers high Accuracy, Precision, Recall, and F1-score for normal/malicious detection when evaluated on four benchmark datasets. We have shown that the proposed framework is an effective solution to the intrusion detection challenge.

While our proposed CAT ensemble model has demonstrated remarkable performance in network intrusion detection, there are several avenues for future research and improvement.

Incorporating Additional Datasets To ensure the robustness and generalizability of CAT, it is essential to evaluate its performance on a broader range of intrusion detection datasets. Including datasets from different network environments and attack types would provide more comprehensive insights into CAT's capabilities and limitations.

Targeting Specific Attack Types Extending CAT to specialize in detecting specific types of attacks could lead to improved accuracy and efficacy. Focusing on attacks that are prevalent in specific industries or applications would allow for tailored intrusion detection solutions with higher detection rates.

Exploring Transformer Variants Transformers have shown remarkable success in NLP tasks. Investigating different transformer variants and architectures could offer additional insights into their effectiveness for network intrusion detection. This exploration could include leveraging pre-trained transformers or using different attention mechanisms.

Enhancing Ensemble Strategies While CAT_O has demonstrated superior performance, investigating novel ensemble strategies, including meta-learning and ensemble selection methods [55], or combining additional classifiers could potentially lead to further improvements. Adaptive ensemble techniques that dynamically adjust classifier weights during runtime based on real-time network conditions could be explored.

Real-Time Implementation The deployment of CAT in real-time network environments is a critical area for future investigation. Optimizing the model's speed and resource efficiency to handle the high-speed and large-scale data flows in real-world networks would be of great practical significance.

Evaluation on New Intrusion Scenarios As network technologies and attack patterns continue to evolve, CAT's performance needs to be assessed on emerging intrusion scenarios. This evaluation would ensure that CAT remains relevant and effective in contemporary network security challenges.

In conclusion, our proposed CAT ensemble model provides a strong foundation for future research in the field of network intrusion detection. By addressing the above aspects, we can further advance the state-of-the-art in intrusion detection systems and develop robust solutions to safeguard networks from ever-evolving cyber threats.

Appendix Acronyms

Ada	Adaboost
AE	Autoencoder
AI	Artificial Intelligence
ANN	Artificial Neural Network
AUC	Area Under the Curve
BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolutional Neural Network
DBN	Deep Belief Network
DDoS	Distributed Denial of Service
DL	Deep Learning
DNN	Deep Neural Network
DT	Decision Tree
ECG	Electrocardiogram
ELM	Extreme Learning Machine
FN	False Negative
FP	False Positive
FRS	Framingham Risk Score
FT	FT-Transformer
GB	Gradient Boosting
IDS	Intrusion Detection System
ID-RDRL	Intrusion Detection based on Reinforcement Learning
KNN	K-Nearest Neighbor
LR	Logistic Regression
LSTM	Long Short-Term Memory
MSE	Mean Squared Error
ML	Machine Learning
MLP	Multi-layer Perceptron
NB	Naive Bayes
NID	Network Intrusion Detection
RF	Random Forest
SDAE	Stacked Denoising Autoencoder
SGD	Stochastic Gradient Descent
SVM	Support Vector Machine
TPE	Tree-structured Parzen Estimator
TP	True Positive
TN	True Negative
VAE	Variational Autoencoder
XGBoost	Extreme Gradient Boosting

Author Contributions Zheng Zhang: Original draft, Validation, Software, Methodology. Amit Das: Draft, Validation, Software, Methodology. Guan Huang: Methodology, Conceptualization. Sanjeev Baskiyar: Supervision, Writing - review & editing, Conceptualization.

Funding This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

Data Availability No datasets were generated or analysed during the current study.

Declarations

Competing interests The authors declare no competing interests.

References

1. Van Eck N, Waltman L (2010) Software survey: vosviewer, a computer program for bibliometric mapping. Scientometrics 84(2):523–538
2. Dong B, Wang X (2016) Comparison deep learning method to traditional methods using for network intrusion detection: In: 2016 8th IEEE international conference on communication software and networks (ICCSN). IEEE, pp 581–585
3. Kuncheva LI (2014) Combining pattern classifiers: methods and algorithms. John Wiley & Sons
4. Lazzarini R, Tianfield H, Charissis V (2023) A stacking ensemble of deep learning models for iot intrusion detection. Knowl-Based Syst 279:110941
5. Kumar P, Gupta GP, Tripathi R, Garg S, Hassan MM (2021) Dltif: deep learning-driven cyber threat intelligence modeling and identification framework in iot-enabled maritime transportation systems. IEEE Trans Intell Trans Syst 24(2):2472–2481
6. Khan F, Jan MA, Alturki R, Alshehri MD, Shah ST, ur Rehman A (2023) A secure ensemble learning-based fog-cloud approach for cyberattack detection in IOMT. IEEE Trans Ind Inf 19(10):10125–10132
7. Gorishniy Y, Rubachev I, Khrulkov V, Babenko A (2021) Revisiting deep learning models for tabular data. Adv Neural Inf Process Syst 34:18932–18943
8. Chen T, He T, Benesty M, Khotilovich V, Tang Y, Cho H, Chen K et al (2015) Xgboost: extreme gradient boosting 0.4-2. R Package Version 1(4):1–4
9. Tang W, Long G, Liu L, Zhou T, Jiang J, Blumenstein M (2020) Rethinking 1d-cnn for time series classification: a stronger baseline. arXiv:2002.10061
10. Liao Y, Vemuri VR (2002) Use of k-nearest neighbor classifier for intrusion detection. Comput Secur 21(5):439–448
11. Lippmann RP, Fried DJ, Graf I, Haines JW, Kendall KR, McClung D, Weber D, Webster SE, Wyschogrod D, Cunningham RK, et al (2000) Evaluating intrusion detection systems: the 1998 darpa off-line intrusion detection evaluation. In: Proceedings DARPA information survivability conference and exposition. DISCEX'00, IEEE, vol 2, pp 12–26
12. Mukkamala S, Janoski G, Sung A (2002) Intrusion detection using neural networks and support vector machines. In: Proceedings of the 2002 international joint conference on neural networks. IJCNN'02 (Cat. No. 02CH37290), IEEE, vol 2, pp 1702–1707
13. Amor NB, Benferhat S, Elouedi Z (2004) Naive bayes vs decision trees in intrusion detection systems. In: Proceedings of the 2004 ACM symposium on applied computing, pp 420–424
14. Tavallaee M, Bagheri E, Lu W, Ghorbani AA (2009) A detailed analysis of the kdd cup 99 data set. In: 2009 IEEE symposium on computational intelligence for security and defense applications. IEEE, pp 1–6
15. Inuwa MM, Das R (2024) A comparative analysis of various machine learning methods for anomaly detection in cyber attacks on iot networks. Int Things 26:101162
16. Jiang K, Wang W, Wang A, Wu H (2020) Network intrusion detection combined hybrid sampling with deep hierarchical network. IEEE Access 8:32464–32476

17. Moustafa N, Slay J (2015) Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In: (2015) military communications and information systems conference (MilCIS). IEEE 2015:1–6
18. Wei P, Li Y, Zhang Z, Hu T, Li Z, Liu D (2019) An optimization method for intrusion detection classification model based on deep belief network. IEEE Access 7:87593–87605
19. Gwon H, Lee C, Keum R, Choi H (2019) Network intrusion detection based on lstm and feature embedding. arXiv:1911.11552
20. Cil AE, Yildiz K, Buldu A (2021) Detection of ddos attacks with feed forward based deep neural network model. Expert Syst Appl 169:114520
21. Aguru AD, Erukala SB (2024) A lightweight multi-vector ddos detection framework for iot-enabled mobile health informatics systems using deep learning. Inf Sci 662:120209
22. Dang Q, Yang S, Liu Q, Ruan J (2024) Adaptive and communication-efficient zeroth-order optimization for distributed internet of things. IEEE Int Things J
23. Vinayakumar R, Alazab M, Soman K, Poornachandran P, Al-Nemrat A, Venkatraman S (2019) Deep learning approach for intelligent intrusion detection system. IEEE Access 7:41525–41550
24. Li Y, Xu Y, Liu Z, Hou H, Zheng Y, Xin Y, Zhao Y, Cui L (2020) Robust detection for network intrusion of industrial iot based on multi-cnn fusion. Measurement 154:107450
25. Liu J, Zhang W, Tang Z, Xie Y, Ma T, Zhang J, Zhang G, Niyoyita JP (2020) Adaptive intrusion detection via ga-gogmm-based pattern learning with fuzzy rough set-based attribute selection. Expert Syst Appl 139:112845
26. Sethi K, Kumar R, Prajapati N, Bera P (2020) Deep reinforcement learning based intrusion detection system for cloud infrastructure. In: 2020 International Conference on COMmunication Systems & NETworkS (COMSNETS), IEEE, pp 1–6
27. Termos M, Ghalmane Z, Fadlallah A, Jaber A, Zghal M et al (2024) Gdlc: a new graph deep learning framework based on centrality measures for intrusion detection in iot networks. Int Things 26:101214
28. Aburomman AA, Reaz MBI (2017) A survey of intrusion detection systems based on ensemble and hybrid classifiers. Comput Secur 65:135–152
29. Zainal A, Maarof MA, Shamsuddin SM et al (2009) Ensemble classifiers for network intrusion detection system. J Inf Assur Secur 4(3):217–225
30. Hsu Y-F, He Z, Tarutani Y, Matsuoka M (2019) Toward an online network intrusion detection system based on ensemble learning. In: 2019 IEEE 12th international conference on cloud computing (CLOUD). IEEE, pp 174–178
31. Das S, Saha S, Priyoti AT, Roy EK, Sheldon FT, Haque A, Shiva S (2021) Network intrusion detection and comparative analysis using ensemble machine learning and feature selection. IEEE Trans Netw Serv Manag 19(4):4821–4833
32. Rajagopal S, Kundapur PP, Hareesha KS (2020) A stacking ensemble for network intrusion detection using heterogeneous datasets. Secur Commun Netw 2020(1):4586875
33. Rajadurai H, Gandhi UD (2022) A stacked ensemble learning model for intrusion detection in wireless network. Neural Comput Appl 34(18):15387–15395
34. Thockchom N, Singh MM, Nandi U (2023) A novel ensemble learning-based model for network intrusion detection. Complex Intell Syst 9(5):5693–5714
35. Moustafa N, Turnbull B, Choo K-KR (2018) An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. IEEE Int Things J 6(3):4815–4830
36. Soleymanzadeh R, Aljasim M, Qadeer MW, Kashef R (2022) Cyberattack and fraud detection using ensemble stacking. AI 3(1):22–36
37. Ravi V, Chaganti R, Alazab M (2022) Recurrent deep learning-based feature fusion ensemble meta-classifier approach for intelligent network intrusion detection system. Comput Electr Eng 102:108156
38. Folino F, Folino G, Guarascio M, Pisani FS, Pontieri L (2021) On learning effective ensembles of deep neural networks for intrusion detection. Inf Fusion 72:48–69
39. Wang Z, Liu J, Sun L (2022) Efs-dnn: an ensemble feature selection-based deep learning approach to network intrusion detection system. Secur Commun Netw 2022(1):2693948
40. Hassini K, Khalis S, Habibi O, Chemmakha M, Lazaar M (2024) An end-to-end learning approach for enhancing intrusion detection in industrial-internet of things. Knowl-Based Syst 294:111785
41. Zohourian A, Dadkhah S, Molyneaux H, Neto ECP, Ghorbani AA (2024) Iot-prids: leveraging packet representations for intrusion detection in iot networks. Comput Secur 146:104034
42. Greenberg MD (1998) Advanced engineering mathematics, Pearson Education India
43. Albawi S, Mohammed TA, Al-Zawi S (2017) Understanding of a convolutional neural network, In: 2017 International conference on engineering and technology (ICET). IEEE, pp 1–6
44. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, Kaiser Ł, Polosukhin I (2017) Attention is all you need. Adv Neural Inf Process Syst 30
45. Lin T, Wang Y, Liu X, Qiu X (2022) A survey of transformers. AI Open
46. Devlin J, Chang M-W, Lee K, Toutanova K (2018) Bert: pre-training of deep bidirectional transformers for language understanding. arXiv:1810.04805
47. Wang Q, Li B, Xiao T, Zhu J, Li C, Wong DF, Chao LS (2019) Learning deep transformer models for machine translation. arXiv:1906.01787
48. Shwartz-Ziv R, Armon A (2022) Tabular data: deep learning is not all you need. Inf Fusion 81:84–90
49. Bergstra J, Bardenet R, Bengio Y, Kégl B (2011) Algorithms for hyper-parameter optimization. Adv Neural Inf Process Syst 24
50. Bay SD, Kibler D, Pazzani MJ, Smyth P (2000) The uci kdd archive of large data sets for data mining research and experimentation. ACM SIGKDD Explor Newslett 2(2):81–85
51. Stolfo J, Fan W, Lee W, Prodromidis A, Chan PK (2000) Cost-based modeling and evaluation for data mining with application to fraud and intrusion detection. Results from the JAM Project by Salvatore pp 1–15
52. Sharafaldin I, Lashkari AH, Ghorbani AA (2018) Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp 1:108–116
53. Neto ECP, Dadkhah S, Ferreira R, Zohourian A, Lu R, Ghorbani AA (2023) Ciciot 2023: a real-time dataset and benchmark for large-scale attacks in iot environment. Sensors 23(13):5941
54. Zhou Z-H (2012) Ensemble methods: foundations and algorithms. CRC press
55. Yang Y, Lv H, Chen N (2023) A survey on ensemble learning under the era of deep learning. Artif Intell Rev 56(6):5545–5589

Zheng Zhang received his Ph.D. in Computer Science and Software Engineering from Auburn University, USA, in 2024. He is an assistant professor in the Department of Computer Science and Information Systems at Murray State University. His research interests include artificial intelligence, natural language processing, intrusion detection, and neural-symbolic AI.

Amit Das received his Ph.D. in Computer Science and Software Engineering from Auburn University, USA, in 2024. He is currently an Assistant Professor in the Department of Computer and Information Systems at the University of North Alabama. His primary research interest includes natural language processing.

Guan Huang is a Ph.D. student in the Department of Computer Science and Software Engineering at Auburn University, where he also earned his M.S. in 2023. His research interests include federated learning, transformer-based architectures, graph neural networks, ensemble learning, model distillation, differentiable pruning, and advanced model-optimization techniques. His work addresses practical challenges in learning under noisy labels, network intrusion detection, cybersecurity analytics, low-Earth-orbit satellite systems, and efficient large-language-model inference via dynamic pruning. He has published in premier venues such as ICML and ICMLA and is a recipient of the Auburn University Graduate Research Fellowship.

Sanjeev Baskiyar is a Full Professor in the Department of Computer Science and Software Engineering at Auburn University, Auburn, Alabama. He received the Ph.D. and M.S.E.E. degrees from the University of Minnesota, Minneapolis, a B.E. degree in Electronics and Communications from the Indian Institute of Science, Bangalore. He has taught courses in Computer Architecture, Real-time and Embedded Computing, Operating Systems, Microprocessor Programming and Interfacing, and VLSI Design. His research interests are in all aspects of Computer Systems Architecture. He has also worked in industry as a Senior Software Engineer at Unisys and a Computer Engineer at Tata Motors.