scientific reports

Check for updates

OPEN

An IoT intrusion detection framework based on feature selection and large language models fine-tuning

Huan Ma[1,2]✉, Wan Zhang[2], Dalong Zhang[1] & Baozhan Chen[3]

The rapid proliferation of Internet of Things (IoT) devices has significantly expanded the network attack surface, necessitating the deployment of advanced AI (artificial intelligence)-based intrusion detection systems (IDS) to bolster IoT security. But existing methods face two significant challenges: (1) Feature redundancy: Current approaches extract numerous flow-level features to learn attack behavior, resulting in high computational complexity and substantial redundant information. (2) Class imbalance: Limited attack traffic samples hinder models from effectively learning attack patterns. However, existing algorithms typically address only one of these issues, overlooking their interconnection. Therefore, we propose a Feature Selection and Large Language Models (LLMs)-based IoT intrusion detection framework (FSLLM). At its core is a multi-stage feature selection algorithm combining Minimum Redundancy Maximum Relevance algorithm (mRMR) and a Pearson Correlation Coefficient (PCC)-improved Covariance Matrix Adaptation Evolution Strategy algorithm (CMA-ES). This algorithm utilizes the CMA-ES algorithm for feature search while also taking into account the mutual information and collinearity among features, thereby more effectively reducing redundancy features. Subsequently, we employ the selected representative features to fine-tune LLMs and generate additional attack samples. This approach effectively reduces the computational cost of fine-tuning while producing higher-quality samples. Furthermore, we employ Focal Loss (FL) function-improved LightGBM as the classifier to improve detection performance. We evaluate our framework on five IoT intrusion detection datasets: NF-ToN-IoT-v2, NF-UNSW-NB15-v2, NF-BoT-IoT-v2, NF-CSE-CIC-IDS2018-v2, and CIC-ToN-IoT. Experimental results demonstrate that FSLLM achieves comparable or superior accuracy to current state-of-the-art methods while reducing redundant features by over 80%.

Keywords Internet of Things (IoT), Intrusion detection, Feature selection, large language models (LLMs), class-imbalance

With the proliferation of IoT services, the interconnection and intercommunication of heterogeneous devices across different platforms have become increasingly facilitated[1]. An analysis conducted in 2023 projected that the global number of IoT devices would increase from 15.1 billion in 2020 to over 29 billion by 2030[2]. However, the widespread adoption of IoT devices has also expanded the attack surface for hackers, rendering these devices primary targets for cyberattacks. IoT networks are particularly susceptible to attacks such as denial of service (DoS), distributed DoS (DDoS), and reconnaissance attacks[3]. The severity of these threats is exemplified by two major incidents: the 2015 cyberattack on Ukraine's power grid, which left over 230,000 people without electricity, and the 2016 Mirai worm-fueled DDoS attack that disrupted IoT devices worldwide[4,5]. Consequently, detecting and preventing cyber threats in IoT networks has emerged as a critical task in the field of cybersecurity.

To enhance IoT security and address emerging threats, AI-based intrusion detection algorithms have been widely adopted[6]. Methods based on graph neural networks[7] and deep learning[8] have currently achieved promising results in the field of intrusion detection. However, due to the significantly lower proportion of complex attack traffic (e.g., "worm attacks" and "shell-code attacks") in IoT scenarios, current algorithms face two major challenges: (1) Feature redundancy: Existing methods extract numerous flow-level features to learn

[1]School of Cyber Science and Engineering, Zhengzhou University, Zhengzhou 450001, China. [2]Software Engineering College, Zhengzhou University of Light Industry, Zhengzhou 450066, China. [3]Ational Telemedicine Center of China, The First Affiliated Hospital of Zhengzhou University, Zhengzhou 450052, China. ✉email: mahuanresearch@126.com

at-tack behaviors, resulting in high computational complexity and redundant information. (2) Class imbalance: Limited attack traffic samples hinder effective model learning of attack patterns.

For feature redundancy, two main approaches are used: feature selection and feature extraction. Feature selection includes wrapper, filter, and embedded methods[9–11], while feature extraction primarily employs unsupervised learning techniques[12,13]. Heuristic algorithm-based feature selection methods[10,11] have gained attention for their ability to search for efficient feature subsets. However, current heuristic-based feature selection methods often simplify feature selection into a single-objective optimization problem, neglecting feature collinearity and correlations with target variables. This limitation hinders effective identification and elimination of redundant features in datasets, potentially affecting model performance and generalization.

For class imbalance, researchers have proposed various methods, including data-level resampling techniques and algorithm-level deep learning approaches. Resampling techniques include Oversampling, undersampling[14], and Synthetic Minority Oversampling Technique (SMOTE) and its variants[15]. However, these techniques may lead to data distribution distortion or overfitting. With the advancement of deep learning, techniques such as Generative Adversarial Networks (GANs)[16] and Variational Autoencoders (VAEs)[17] have been used to generate minority class samples. These methods attempt to generate more realistic samples by learning the latent distribution of data. However, they often struggle with high-dimensional tabular data and may produce inconsistent sample quality.

Previous research reveals that current methods still have many shortcomings, and many approaches focus on either feature redundancy or class imbalance without considering their interrelationship. However, by using feature selection algorithms to select representative feature subsets, we can obtain higher-quality feature sets, effectively aiding data augmentation algorithms in reducing training costs and generating higher-quality samples. In this study, we propose FSLLM, an IoT intrusion detection framework that integrates multi-stage feature selection and data augmentation. Our approach leverages a novel MCP feature selection algorithm to reduce redundancy and improve feature representation. Furthermore, we fine-tune LLMs using these selected features to generate high-quality synthetic samples, thereby mitigating class imbalance. Finally, we employ a lightweight LightGBM classifier with an FL function to enhance detection performance in imbalanced datasets. The main contributions of this paper are as follows:

1. We have developed a multi-stage feature selection algorithm (MCP feature selection algorithm) that considers both feature collinearity and redundancy, significantly reducing the number of features while maintaining model performance.
2. By fine-tuning LLMs with representative features selected through our algorithm, we achieved high-quality minority class sample generation while reducing fine-tuning costs. This approach offers a novel perspective on addressing class imbalance in the field of intrusion detection.
3. We have employed the lightweight LightGBM as a classifier and introduced FL function to optimize its performance on imbalanced datasets.
4. We have conducted experiments on five recent IoT intrusion detection datasets: NF-CSE-CIC-IDS2018-v2, NF-ToN-IoT-v2, NF-UNSW-NB15-v2, NF-BoT-IoT-v2, and CIC-ToN-IoT. These experiments validated the generalizability and feasibility of our proposed method.

The remainder of this paper is organized as follows. "Related work" section briefly reviews related research on feature selection and data augmentation applied to IoT intrusion detection. "Methodology" section introduces our framework, focusing on the three components illustrated in Fig. 1. "Results and discussion" section evaluates the proposed framework in various environments and presents the experimental results through detailed analysis. Finally, we conclude this study and propose future work directions in "Conclusions" section.

Related work

AI-based IoT intrusion detection

In the ?eld of AI-based NIDS, many studies have been conducted to apply machine learning and deep learning technologies to anomaly detection. Sarhan et al.[12] employed Principal Component Analysis (PCA), Auto-encoder (AE), and Linear Discriminant Analysis (LDA) for feature extraction on UNSW-NB15, ToN-IoT, and CSE-CIC-IDS2018 datasets, followed by evaluations using Deep Feed Forward (DFF), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Decision Tree (DT), Logistic Regression (LR), and Naive Bayes (NB) models. Yang et al.[13] utilized stacked sparse autoencoder (SSAE) for feature extraction and temporal convolutional network (TCN) for IoT attack detection. The approach reduces training time and resource demands by over 50% while maintaining detection accuracy. Majhi et al.[18] proposed an IoT intrusion detection algorithm using LightGBM, optimized with the Grasshopper Optimization Algorithm (GOA), achieving a 97% detection success rate on the NF-UNSW-NB15 dataset.

In contrast, Shaker et al.[19] explored deep learning models: CNNs, Deep Neural Networks (DNNs), RNNs on NF-UNSW-NB15, NF-BoT-IoT, and NF-ToN-IoT datasets, finding DNNs performed best in binary classification with 98.74% accuracy. However, all models showed moderate performance in multi-class classification. To leverage long-term behavior recognition, Manocchio et al.[20] framed traffic classification as a text classification task, using Transformer-based architectures like GPT-2 and BERT. GPT-2 performing best on NSL-KDD, CSE-CIC-IDS, and UNSW-NB15 datasets. Karthikeyan et al.[21] introduced the FA-ML technique, combining machine learning with the Firefly Algorithm, to enhance intrusion detection in IoT systems. Using a support vector machine (SVM) model with parameter tuning via the Grey Wolf Optimizer (GWO), the FA-ML method achieves a high accuracy of 99.34% on the NSL-KDD dataset.

Termos et al.[7] presented the Graph Deep Learning framework based on Centrality measures (GDLC) to improve intrusion detection in IoT networks by dynamically selecting centrality measures and integrating them

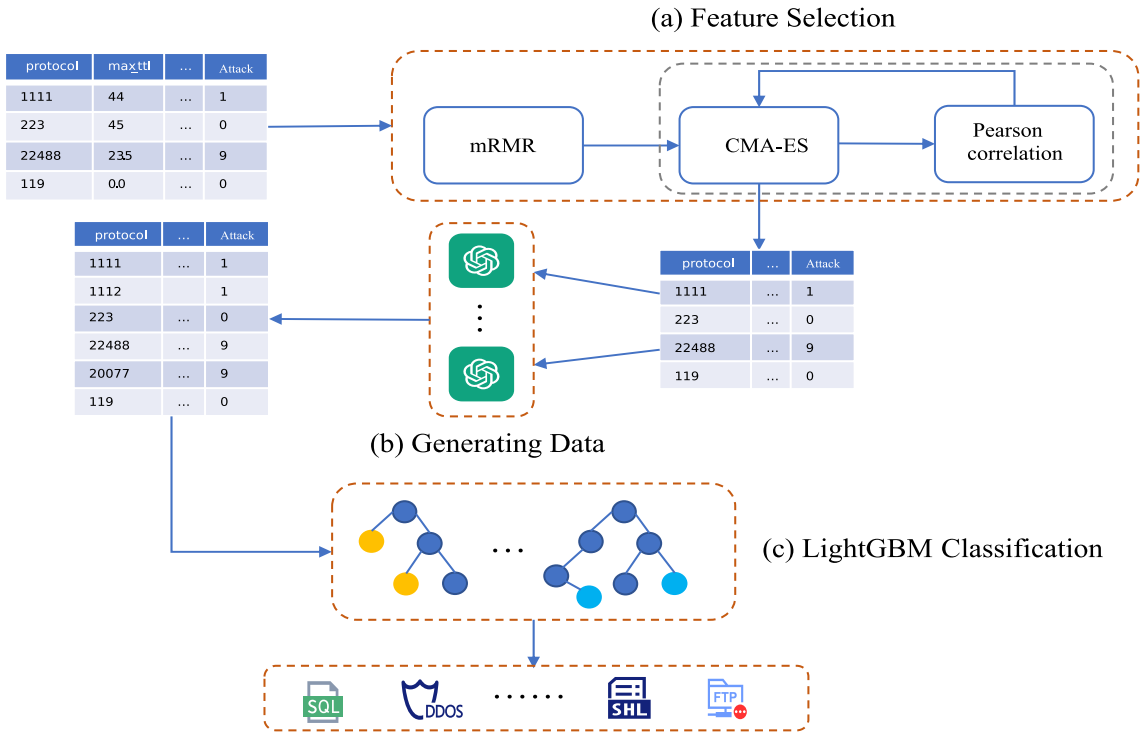

Fig. 1. FSLLM framework for IoT intrusion detection: (**a**) The feature selection algorithm identifies essential features from IoT intrusion detection traffic. (**b**) LLMs are fine-tuned using a dataset constructed from these critical features, allowing the model to generate new data samples. (**c**) The LightGBM classifier is improved with the FL function.

with deep learning models like CNNs, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU). The approach tested on various datasets, shows a detection rate improvement of up to 7.7%. Nguyen et al.[22] and Wang et al.[23] both focused on graph-based approaches. Nguyen's self-supervised graph neural network achieved 99% accuracy in binary classification by enhancing graph representations through auxiliary attribute learning. Similarly, Wang's graph attention networks, which considered both state and temporal features, also achieved 99% bi-nary classification accuracy on the NF-ToN-IoT-v2 dataset.

Feature selection algorithms in IoT intrusion detection

Currently, feature selection algorithms for IoT intrusion detection have also been extensively researched, effectively enhancing models' interpretability and prediction speed. Sarhan et al.[9] conducted feature selection experiments on six IoT intrusion detection datasets using chi-square test, Information Gain (IG), and PCC methods. For UNSW-NB15, Random Forest (RF) achieved 98.62% and 98% accuracy on original and NetFlow versions, using 7 and 3 features respectively. On ToN-IoT, RF reached 97.49% accuracy with 20 features on the original dataset and 99.38% with 6 features on the NetFlow version. For CSE-CIC-IDS2018, RF attained 98.36% and 95.51% accuracy on original and NetFlow versions, using 3 and 6 features respectively. Leevy et al.[24] evaluated the impact of ensemble feature selection techniques (FSTs) on detecting IoT attacks using the Bot-IoT dataset. They employed IG, IG ratio, and Chi-squared as feature selection methods, experimenting with four ensemble learners and four non-ensemble learners. The study found that the method reduced computational burden by decreasing the number of features.

Mohy-Eddine et al.[25] combined Isolation Forest (IF) and PCC for feature selection. They employed two strategies: (1) using IF to remove anomalies followed by PCC feature selection, and (2) applying PCC feature selection before removing anomalies. Their method achieved accuracies of 99.30% and 99.18% on the NF-UNSW-NB15-v2 dataset. Komisarek et al.[26] used chi-square test, RF feature importance, and Lasso L1 to select ten crucial features from five Net-Flow datasets.

Amin et al.[27] proposed the Chaotic Zebra Optimization Algorithm (CZOA) for feature selection, achieving 99.83% accuracy with LSTM on CSE-CIC-IDS2018. Khammassi et al.[10] utilized Genetic Algorithms (GA) and logistic regression(LR) for feature selection, achieving 99% accuracy on the KDD99 dataset and 81.42% accuracy on the UNSW-NB15 dataset. Subramani et al.[11] introduced a rule-based and multi-objective PSO algorithm for feature selection, combined with an improved SVM for classification. This approach demonstrated reduced training and prediction times while achieving strong results on three public datasets.

Traffic data augmentation algorithms for IoT intrusion detection

Due to the complexity and diversity of IoT attacks, collecting attack traffic during the data acquisition process is often challenging. Consequently, many researchers have attempted to generate attack traffic using data

augmentation algorithms. Talukder et al.[14] proposed a novel ML-based network intrusion detection model using Random Oversampling, Stacking Feature Embedding based on clustering results, and PCA for dimension reduction. Their model achieved high accuracy rates on UNSW-NB15 (99.59% for RF, 99.95% for Extra Trees), CIC-IDS-2017 (99.99% for RF), and CIC-IDS-2018 (99.94% for Decision Tree and RF) datasets. sayegh et al.[15] developed an LSTM-based intrusion detection system for IoT networks, incorporating SMOTE to address data imbalance. Their model outperformed existing methodologies on CICIDS2017, NSL-KDD, and UNSW-NB15 datasets, demonstrating improved accuracy in detecting network intrusions and contributing to enhanced IoT security. Mouiti et al.[28] employed Adaptive Synthetic Sampling (ADASYN) for Oversampling, achieving 99% binary classification accuracy on UNSW-NB15. Similarly, Liu et al.[29] combined ADASYN with LightGBM, obtaining accuracies of 92.57%, 89.56%, and 99.91% on NSL-KDD, UNSW-NB15, and CI-CIDS2017 datasets.

Ding et al.[16] utilized a hybrid model with Conditional Generative Adversarial Network (CGAN), Deep AE, and RF, achieving 99.8% binary and 99.6% multi-class classification accuracy. Soflaei et al.[30] addressed data imbalance in UNSW-NB15 with Conditional Tabular Generative Adversarial Network (CTGAN), achieving 98% binary and 95% multi-class classification accuracy using XGBoost. Wang et al.[31] combined WGAN-gp with graph neural networks, achieving 93.7% binary classification accuracy on NF-BoT-IoT using E-GraphSAGE. Li et al.[32] introduced a CGAN and BERT-based model, enhancing data generation quality and achieving optimal performance on NF-ToN-IoT-v2, CSE-CIC-IDS2018, and NF-UNSW-NB15-v2.

Methodology

As shown in Fig. 1, the core of the FSLLM framework is a multi-stage feature selection algorithm, which we call the MCP algorithm. This algorithm comprises two main steps:

1. Redundant Feature Elimination: We first assess the redundancy between features and their relevance to the target variable using the mRMR algorithm[33]. The mRMR algorithm is a widely used feature selection method that evaluates both feature redundancy and feature importance. It ensures that selected features are not only highly relevant to the target variable but also minimally redundant among themselves. To achieve this, we use mutual information to measure dependencies between features, helping to reduce the initial search space for subsequent steps.
2. Feature Search Optimization: After the initial filtering, we refine the feature selection process using the CMA-ES algorithm[34]. CMA-ES is an evolutionary black-box optimization algorithm that is particularly effective for complex, high-dimensional optimization problems. Since CMA-ES operates in a continuous space, we employ a continuous relaxation encoding mechanism to map its solutions to a binary feature selection space. Furthermore, to mitigate feature redundancy caused by collinearity, we incorporate an adaptive search strategy that dynamically adjusts the PCC threshold during the optimization process. The PCC is used to measure the linear correlation between variables and helps in refining feature selection by ensuring that highly correlated features do not introduce redundancy.

Following feature selection, FSLLM constructs a fine-tuning dataset using the selected key features and addresses class imbalance by generating additional samples for underrepresented attack traffic. The quality of the generated samples improves due to the more representative features obtained during feature selection.

After feature selection and data augmentation, FSLLM employs FL-improved LightGBM as the classifier to enhance detection performance. LightGBM is a tree-based machine learning algorithm that leverages gradient boosting to improve classification accuracy while maintaining low computational requirements, making it suitable for real-world deployment scenarios.

MCP feature selection algorithm

The purpose of feature selection for IoT intrusion detection datasets is to eliminate redundant features in the original dataset while maintaining a relatively high prediction performance, thereby enhancing the model' s interpretability, training efficiency, and prediction speed. Formally, let' s consider an IoT intrusion detection dataset $\mathcal{D} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where $x_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ represents a p -dimensional feature vector, and y_i is the corresponding label. The objective of feature selection is to select a subset $S \subseteq \{1, \ldots, n\}$ from the p features to minimize the number of features while keeping the loss in model performance as low as feasible.

To better address the feature redundancy problem, this paper proposes a feature selection algorithm named MCP. The detailed implementation of the MCP feature selection algorithm is presented in Algorithm 1. Firstly, the algorithm determines the minimum threshold for subsequent calculation of collinear features by computing the PCC between features of the training set. For the dataset, we calculate the PCC between each pair of features x and y and subsequently identify the most frequently occurring value as the minimum threshold for collinear feature calculation.

Input: train_data, train_label, valid_data, valid_label, epoch, max_gen, population_size
Output: feature_subset

base_corr_ratio ← **find_base_corr_ratio**(train_data)
features ← **Obtain the initial feature subset from the mRMR algorithm**
optimizer ← **Initialize the CMA-ES algorithm**
for $i = 1$ **to** epoch **do**
    train_data ← train_data[features]
    valid_data ← valid_data[features]
    for $j = 1$ **to** max_gen **do**
        solutions ← []
        for $k = 1$ **to** population_size **do**
            x_eval, x_tell ← **Get candidate solutions from optimizer**
            x_eval ← **sigmoid(x_eval)**
            x_eval ← [1 if $x > 0.5$ else 0 for x in x_eval]
            fitness_score ← **fitness_function**(x_eval, train_data, valid_data, train_label, valid_label)
            solutions.add(x_tell, fitness_score)
        end for
        update_optimizer(solutions)
    end for
    current_features ← **Get best solutions from optimizer**
    corr_ratio ← **Calculate corr_ratio by base_corr_ratio**
    feature_subset ← **remove_collinear_features**(current_features, corr_ratio)
    if feature_subset == current_features **then**
        break
    end if
    features ← feature_subset
end for
return feature_subset

Algorithm 1. MCP feature selection algorithm.

Redundant feature elimination (mRMR)
In the feature selection phase, we initially employ the Fast-mRMR[35] (An improved version of the mRMR algorithm, accelerated by introducing GPU support for computation.) algorithm to obtain a preliminary feature subset. The mRMR algorithm aims to select a subset from a given feature set by computing the mutual information between features and the target variable, as well as among the features themselves. Its goal is to maximize the relevance of the selected features to the target variable while minimizing redundancy within the subset. Mutual information serves as a metric to quantify the interdependence between two random variables. For variables X and Y, their mutual information can be calculated using equation (1), where $p(x, y)$ represents the joint probability distribution of variables X and Y, and $p(x)$ and $p(y)$ are the marginal probability distributions of X and Y. This formula measures the dependency between variables. If X and Y are completely independent, then $I(X; Y) = 0$. For example, suppose a dataset contains two variables: traffic size (bytes) and attack type (e.g., DoS attack). If attack traffic is typically larger in size, then these two variables will have a high mutual information value.

$$I(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) log \frac{p(x,y)}{p(x)p(y)}. \tag{1}$$

We assume the IoT intrusion detection dataset' s feature set is S and the target variable is c. To achieve the objectives above, the mRMR algorithm conducts feature selection through the following two steps:

1. Maximize the mutual information between features and the target variable: Select features that have a high correlation with the target variable, i.e., $\max_{f_i \in S} I(f_i; c)$.
2. Reduce redundancy among features to ensure that the mutual information between features is minimized, i.e., $\min_{f_i, f_j \in S} \frac{1}{|S|^2} \sum_{f_i, f_j \in S} I(f_i; f_j)$.

Combining the two objectives mentioned above, the selection process of the mRMR algorithm can be defined as equation (2), where $I(f_i; c)$ signifies the mutual information between the feature f_i and the target variable c, and the term $\frac{1}{|S|} \sum_{f_j \in S} I(f_i; f_j)$ calculates the average mutual information between f_i and all features within

the already selected feature set S. The goal of this formula is to select features that are most relevant to the target variable while being least similar to other features. For example, if the feature set includes "source IP address" and "device ID", these two may be highly correlated (as a device typically retains the same IP address). In this case, the mRMR algorithm may remove one of them.

$$max_{f_i \in S} \left[I(f_i; c) - \frac{1}{|S|} \sum_{f_j \in S} I(f_i; f_j) \right]. \tag{2}$$

Feature search optimization (CMA-ES + PCC)
The mRMR algorithm primarily employs linear mutual information measures, which limits its ability to capture complex non-linear relationships and does not address the feature redundancy problem caused by feature collinearity. Therefore, we use the feature subset obtained from the mRMR algorithm as the initial solution for the second phase and introduce an improved version of the CMA-ES algorithm to continue feature selection. Specific improvements include:

1. Pearson Correlation Coefficient: During the CMA-ES iterative search process, we introduce PCC to remove collinear features and provide new initial solutions for subsequent searches.
2. Iterative Search Strategy: We dynamically adjust the PCC threshold for removing collinear features during the iterative search process.
3. Continuous Relaxation-based Feature Encoding: In the feature selection problem, we usually represent feature selection as a binary vector[27]. Each candidate solution is represented by a binary vector $\mathrm{z} = (z_1, z_2, \ldots, z_d)$ of length d, where $z_i \in \{0, 1\}$. Specifically, $z_i = 1$ indicates that the corresponding feature is selected. However, binary encoding cannot express continuous information and cannot be directly applied in the CMA-ES algorithm. Therefore, We employ a sigmoid function for feature encoding to transform the continuous optimization problem into a binary feature selection problem.

During the algorithm' s execution, the mean vector m_0, covariance matrix C_0, and step size σ_0 is first initialized. Specifically, m_0 is a vector of length d, typically initialized with small random values uniformly distributed in the range [-1, 1]. This can be represented as: $m_0 = uniform(-1, 1, d)$. This initialization approach provides a good starting point for the CMA-ES algorithm to explore the search space effectively. The covariance matrix C_0 is usually initialized as an identity matrix I_d of size $d \times d$. The step size σ_0 controls the size of the search range and is commonly initialized to 1.

Subsequently, for each generation t, λ candidate solutions x_i are generated, where $i = 1, 2, \ldots, \lambda$. Assuming the current mean vector is m_t and the step size is σ_t, each candidate solution x_i can be calculated using equation (3).

$$x_i = m_t + \sigma_t \cdot B_t \cdot D_i. \tag{3}$$

B_t is a transformation matrix computed based on the current covariance matrix C_t, which regulates the distribution direction and scope of the candidate solutions x_i. In the computation process, the covariance matrix C_t is first subjected to eigendecomposition to yield its eigenvalues λ_i and corresponding eigenvectors v_i, indexed by i from 1 to d. Subsequently, guided by the effective selection count μ_{eff}($\mu_{eff} \approx \lambda/2$), the initial μ_{eff} eigenvectors of the covariance matrix are selected, and B_t is subsequently computed utilizing equation (4).

$$B_t = \sum_{i=1}^{\mu_{eff}} \sqrt{\lambda_i} \cdot \mathrm{v}_i \cdot \mathrm{v}_i^T. \tag{4}$$

D_i is a d-dimensional standard average random vector introduced to incorporate randomness. It is computed as $D_i \sim \mathcal{N}(0, I_d)$, meaning that D_i is a multivariate average random vector with a mean of zero and a covariance matrix equal to the identity matrix I_d.

Subsequently, for each generated candidate solution x_i, we first transform the feature selection problem from a continuous space to a discrete space. This is achieved using the following formula: $z_i = 1/(1 + e^{-x_i})$, where $z_i \in [0, 1]$ indicates the probability of selecting feature i. Discretization is achieved by applying a threshold value (e.g., $z_i \geq 0.5$ indicates that the feature is selected). This continuous relaxation-based feature encoding approach not only captures the continuous information during the feature selection process but also leverages the strengths of the CMA-ES algorithm in handling continuous optimization problems more effectively. Subsequently, the fitness value $f(x_i)$ is calculated using a fitness function for each candidate solution. Based on the fitness values, the mean vector m_{t+1}, the covariance matrix C_{t+1}, and the step size σ_{t+1} are updated. In this paper, the fitness value is the F1 score (refer to equation (15)) of the LightGBM model.

For the mean vector m_{t+1}, we typically employ equation (5) for the solution. This formula represents the center point (mean) of the next-generation population, which is obtained by updating the previous generation's center point m_t with an adjustment term based on the best-performing individuals. Here, μ represents the number of elites selected, ω_i is the weight assigned when the fitness value is $f(x_i)$, often calculated as $log((\lambda + 1)/2) - log(i)$, and $d_i = x_i - m_t$ represents the centralized candidate solution.

$$m_{t+1} = m_t + \frac{1}{\mu}\sum_{i=1}^{\mu} w_i \cdot d_i. \tag{5}$$

The covariance matrix C_{t+1} can be calculated using equation (6). In this equation, c_1 and c_μ are two control parameters, c_1 is typically used to control the update of the step size, while c_μ is used to control the update of the covariance matrix. $(1 - c_1 - c_\mu) \cdot C_t$ retains a portion of the current covariance matrix to maintain search stability. $c_1 \cdot \sum_{i=1}^{\mu} h_i \cdot d_i d_i^T$ updates the covariance matrix using the outer product of samples, contributing to step size adjustment. $c_\mu \cdot C_t$ further adjusts the covariance matrix to adapt to the new search direction. (typically c_1 = 0.1, c_μ=0.2).

$$C_{t+1} = (1 - c_1 - c_\mu) \cdot C_t + c_1 \cdot \sum_{i=1}^{\mu} h_i \cdot d_i d_i^T + c_\mu \cdot C_t. \tag{6}$$

The step size σ_{t+1} can be calculated using equation (7). In this equation, c_σ is the step size update parameter, typically a small positive number. $E\left[\| N(0,I) \|\right]$ is the expected value of the standard normal distribution. d_{step} is a scaling factor employed to regulate the magnitude of the evolution path length $\| p_\sigma \|$, during the process of updating the step size. This formula controls the search step size σ_t, ensuring that the search is neither too large (which would introduce excessive randomness) nor too small (which could lead to premature convergence to a local optimum).

$$\sigma_{t+1} = \sigma_t \cdot exp\left(\frac{c_\sigma}{d_{\text{step}}}\left(\frac{\| p_\sigma \|}{E[\| N(0,I) \|]} - 1\right)\right). \tag{7}$$

The evolution path p_σ is a vector used to record the changes in the mean vector m_t over the past generations. It can be calculated using equation (8). In this equation, c_{path} is a parameter used to control the update speed of the evolution path.

$$p_\sigma = (1 - c_{path}) \cdot p_\sigma + \sqrt{1 - (1 - c_{path})^2} \cdot \frac{B_t \cdot (m_{t+1} - m_t)}{\sigma_t}. \tag{8}$$

In the CMA-ES algorithm, the mean vector m and covariance matrix C are adaptively updated to explore the search space and identify optimal feature subsets. Imagine CMA-ES searching for the optimal feature subset in a two-dimensional space. The search range in each generation is determined by the covariance matrix C_t. If certain directions correspond to more valuable features, the next generation of the search will be more inclined to explore those directions. However, CMA-ES does not account for feature collinearity, which may result in the inclusion of redundant features. To address this limitation, we propose an iterative search strategy that integrates PCC to assess and mitigate feature collinearity.

Input: data, corr_ratio
Output: selected_features

removed_cols ← empty set
corr_matrix ← **compute correlation matrix from data**
for each col **in** corr_matrix **do**
    if col ∉ removed_cols **then**
        corr_cols ← **find cols most correlated with col in corr_matrix by corr_ratio**
        most_corr ← **column that is most correlated with label among corr_cols**
        if most_corr.size > 1 **then**
            arbitrarily choose one column from most_corr and keep to selected_features
        else
            keep most_corr
        end if
        remove other cols in corr_cols from data and add them to removed_cols
    else
        continue
    end if
end for
return selected_features

Algorithm 2. Remove collinear features.

The specific implementation for removing feature collinearity is presented in Algorithm 2. Let $\mathrm{X} = [\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_n]$ denote the feature matrix, where each x_i represents a feature vector. We first compute the correlation matrix R for the features using PCC. Features with a correlation coefficient $|\mathrm{R}_{ij}|$ exceeding a predefined threshold τ are evaluated. In this process, features exhibiting high collinearity with others are removed, while retaining features that show a stronger correlation with the target variable y. Specifically, if $|corr(\mathrm{x}_i, \mathrm{y})| > |corr(\mathrm{x}_j, \mathrm{y})|$, feature x_i is preserved.

The resulting subset of features, which has been pruned to reduce collinearity, forms the initial population for the CMA-ES algorithm. During optimization, the threshold τ is dynamically adjusted in each iteration t as follows:

$$\tau_t = \tau_0 \times \left(\frac{\tau_{\min}}{\tau_0}\right)^{\frac{t}{T}}. \tag{9}$$

where τ_0 is the initial threshold, $\tau_{\min}$ is the minimum threshold, and T represents the total number of iterations. As iterations proceed, the threshold τ_t is reduced to refine feature selection, ensuring that the final feature subset balances relevance and redundancy effectively.

Improved lightGBM based on FL function

LightGBM[36] typically uses logarithmic and cross-entropy loss functions for binary and multi-class classification tasks, respectively. Although these original loss functions perform well in many cases, they are less effective on imbalanced datasets. To address this limitation, we incorporate FL[37] into the LightGBM framework. In IoT intrusion detection datasets, the binary classification scenario often has relatively balanced positive and negative samples. Therefore, this paper focuses primarily on multi-class classification.

For binary classification, FL is defined as $FL(p_t) = -\alpha_t(1-p_t)^{\gamma} log(p_t)$. where p_t is the probability of predicting the true class, α_t is the class weight used to balance the imbalance between positive and negative samples, and γ is a tuning parameter that controls the rate at which the weight assigned to well-classified examples decreases. And LightGBM builds decision trees using information based on the gradient and Hessian. FL reduces the weight of easily classified samples, thereby emphasizing hard-to-classify samples. When p_t is close to 1 (indicating an easily classified sample), $(1-p_t)^{\gamma}$ approaches 0, reducing its contribution to the loss. Conversely, when p_t is close to 0 (indicating a hard-to-classify sample), $(1-p_t)^{\gamma}$ approaches 1, increasing its contribution to the loss.

As shown in equation (10), to optimize model performance, we determine the initialization score by minimizing the overall FL. In this equation, $\sigma(\cdot)$ represents the sigmoid function, b denotes the initialization score, and y_i is the true label.

$$b^* = \arg\min_b \sum_i FL(\sigma(b), y_i) \tag{10}$$

To extend binary FL to multi-class problems, we use the One-vs-Rest (OvR) strategy[38]. For a problem with K classes, we train K binary classifiers, with each classifier f_i having the decision function $f_i(x) = \sigma(g_i(x) + b_i)$, where $g_i(x)$ is the output of the LightGBM model, and b_i is the initial score for the respective class.

In multi-class tasks, accurate probability distribution requires calibrating the outputs of multiple binary classifiers. Calibration is applied during the prediction phase to ensure the probability distribution is reasonable. This is achieved through softmax normalization, converting each classifier' s scores into relative probabilities:

$$P(y=i|x) = \frac{\exp(f_i(x))}{\sum_j \exp(f_j(x))} \tag{11}$$

This calibration ensures that the sum of the probabilities is 1, allowing the model to better assess the likelihood of a sample belonging to each class, even with imbalanced distributions. After probability calibration, the final multi-class prediction is based on the argmax principle, where $y_{pred} = arg\max_i P(y=i|x)$, thus ensuring that the class with the highest probability is chosen as the prediction.

During LightGBM' s optimization process, we use the derived first and second derivatives to guide the tree growth[36]. In each iteration, the model adjusts the tree structure based on the current gradient and second derivative information to minimize FL.

Traffic data augmentation algorithm based on LLMs

LLMs are deep learning-based AI systems capable of understanding and generating natural language. Their applications span various domains, including natural language processing, dialogue systems, content creation, and information retrieval[39]. In recent years, LLMs have shown particular promise in the field of data augmentation[40]. Unlike traditional methods, LLMs offer exceptional flexibility and precision in generating complex synthetic data. This paper proposes an IoT intrusion detection traffic augmentation method based on fine-tuned LLMs. By fine-tuning on limited attack traffic samples, LLMs can capture and replicate unique features and patterns, generating additional synthetic attack data. This augmented dataset potentially enhances model accuracy and reliability in identifying rare attack types. However, given the complexity and redundancy of IoT traffic data features, directly fine-tuning on all features may lead to inefficient model processing, excessive memory consumption, and increased overfitting risk. This is because the model might focus on irrelevant features while overlooking truly important ones. Therefore, in this study, we employ the MCP feature selection algorithm to

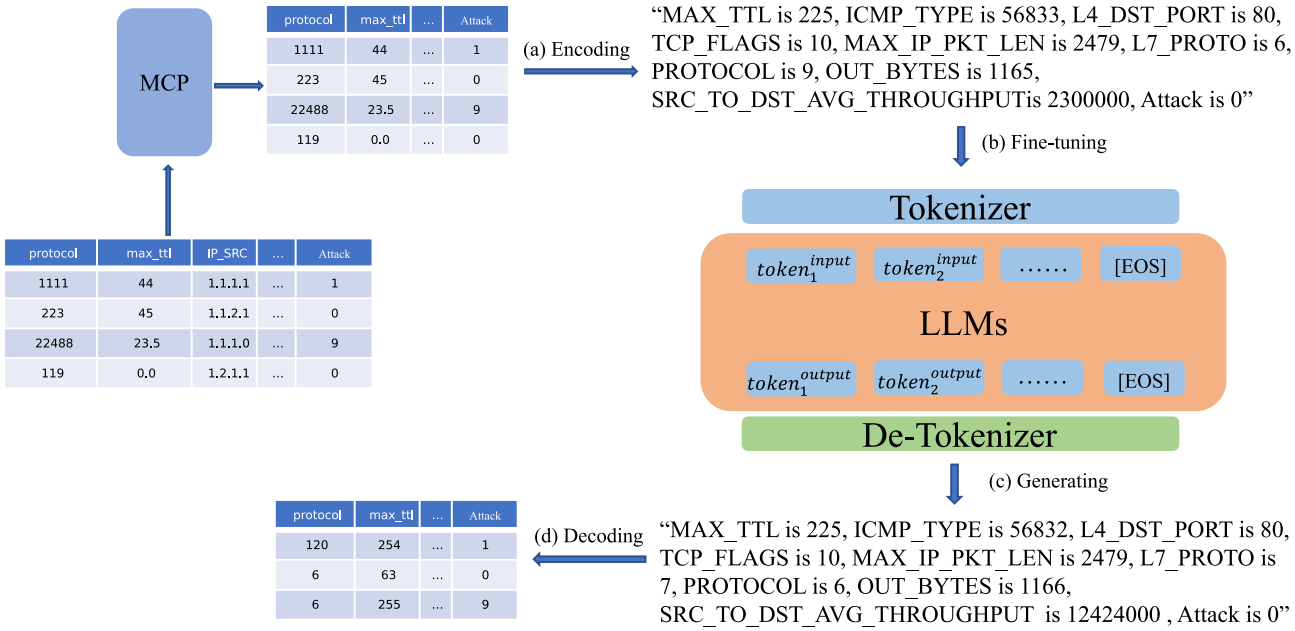

Fig. 2. Augmenting IoT intrusion detection traffic data using LLMs. (**a**) Convert statistical features extracted from raw traffic into long texts with semantic information. (**b**) Fine-tune the LLMs model using the dataset of these long texts. (**c**) Generate similar semantic long texts with the fine-tuned LLMs. (**d**) Decode the generated long texts back into their original statistical features.

identify the most relevant features and construct the fine-tuning dataset. This approach enables the model to better capture key patterns in IoT traffic, thereby improving the efficiency of the fine-tuning process. Moreover, utilizing more representative features facilitates the generation of high-quality samples.

Our approach consists of four stages: (1) Text encoding (Fig. 2a). (2) Fine-tuning the model using the encoded text (Fig. 2b). (3) Generating text based on the fine-tuned model (Fig. 2c). (4) Converting the generated text into IoT intrusion detection traffic data (Fig. 2d). The features used for text encoding in the first phase are selected from the original feature set using the MCP feature selection algorithm.

This ensures efficient and effective text encoding by selecting the optimal feature subset. The subset includes continuous variables (e.g., received traffic value is 59000) and categorical variables (e.g., port number is 80). We represent original features with descriptive phrases such as ' port is 80 ' or ' protocol is 234', maintaining data integrity and clarity throughout the encoding process.

For a feature set $z = (z_1, z_2, \ldots, z_d)$ in an IoT intrusion detection dataset with n rows of samples, each feature z_i in z can be encoded into text using the rule $z_{text} = [z_i, \text{“}is\text{”}, value_{i,j}, \text{“,”}]$, where $value_{i,j}$ denotes the value of the i-th feature in the j-th row. This encoding concatenates each feature description with ", to form coherent statements across the dataset. To enhance the semantic clarity, a comprehensive description is added to each sentence: ' The malicious traffic features and Attack type are described as follows, MAX_TTL is 225, ICMP_TYPE is 56833, L4_DST_PORT is 80, TCP_FLAGS is 10, ..., Attack is 1' This semantic description provides additional context to the encoded features, enhancing the interpretability of the generated text.

During the fine-tuning stage, IoT traffic data is transformed into text enriched with semantic information, thereby converting the IoT intrusion detection traffic data augmentation task into a text generation task. For this purpose, the encoded content z_{text} is tokenized into a sequence $(w_1, w_2, w_3, ..., w_n)$ using a $Tokenizer(t)$ function, where $w \in W$, a predefined vocabulary[41].

In language modeling, the primary objective is to estimate the probability of a given sequence, as shown in equation (12). This probability is often decomposed in an autoregressive manner, represented as $p(w_k|w_1, w_2, \ldots, w_{k-1})$, indicating the likelihood of predicting word $w_1, w_2, \ldots, w_{k-1}$ given all preceding words. The product $\prod_{k=1}^{j}$ signifies that the overall probability of the sequence t is the product of these conditional probabilities for each word. By accurately estimating these probabilities, the model can effectively generate coherent and contextually related sequences of words.

$$p(t) = p(w_1, w_2, \ldots, w_n) = \prod_{k=1}^{j} p(w_k|w_1, w_2, \ldots, w_{k-1}) \tag{12}$$

During the training stage, the model optimizes its parameters to maximize the joint probability of all sequences in z_{text}, which is represented as $\prod_{t \in z_{text}} p(t)$. This objective is commonly achieved using the negative log-likelihood loss, as depicted in equation (13). By taking the negative logarithm of the joint probability, the

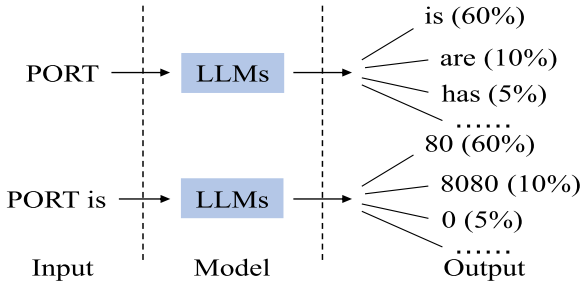

Fig. 3. The sampling process of LLMs.

product is transformed into a sum, and the negative sign ensures that minimizing the loss function is equivalent to maximizing the joint probability. This approach effectively guides the model to achieve the best fit for the data.

$$Loss(t) = -logp(t) = -\sum_{k=1}^{n} logp(w_k|w_1, w_2, \ldots, w_{k-1}) \tag{13}$$

During the sampling stage, given a context (historical sequence) $w_1, w_2, \ldots, w_{k-1}$, a trained model outputs a probability distribution $p\,(w_k|w_1, w_2, \ldots, w_{k-1})$ representing the likelihood of the next token w_k. As illustrated in Fig. 3, different inputs can lead to varied output distributions. For instance, given the input "PORT", the model might predict "is" with 60% probability, "are" with 10%, and "has" with 5%. Similarly, for the input "PORT is", the model might predict "80" with 60% probability, "8080" with 10%, and "0" with 5%. In this paper, we initialize the sequence with ' The malicious traffic features and Attack type are described as follows'.

Various sampling strategies are employed to select the next token from this distribution, including random sampling, greedy search, temperature sampling, top-k sampling, and top-p sampling[42].

In this study, we employ temperature sampling, which adjusts the probability distribution by introducing a temperature parameter T to control the randomness of sampling. Specifically:

1. Adjust the original probability distribution to $p(w_k|w_1, \ldots, w_{k-1})^{1/T}$.
2. Normalize the adjusted probabilities.
3. Perform random sampling based on the normalized distribution.

The temperature T influences the smoothness of the distribution, with higher values increasing randomness and lower values favoring more deterministic selections.

Results and discussion

In this chapter, we first conduct experiments on five separate datasets, using the MCP feature selection algorithm and the data augmentation algorithm based on fine-tuning LLMs to validate the effectiveness of our proposed methods. Subsequently, we compare the final experimental results of the FSLLM framework with state-of-the-art IoT intrusion detection algorithms to demonstrate the framework's efficacy. Finally, we analyze the efficiency of the FSLLM framework to verify its high performance.

Datasets

This study employs five recent IoT intrusion detection datasets[43,44]: NF-CSE-CIC-IDS2018-v2, NF-ToN-IoT-v2, NF-UNSW-NB15-v2, NF-BoT-IoT-v2, and CIC-ToN-IoT. The first four datasets utilize NetFlow for feature extraction, each com-prising 43 features. The last dataset, CIC-ToN-IoT, uses CICFlowMeter for feature selection, containing 83 features. The distribution of attack types across the five datasets is presented in Table 1. These datasets reflect the latest trends in IoT intrusion detection, and thus were selected to evaluate the generalization capability and performance of the FSLLM framework. During the experimental phase, each dataset was initially partitioned into training and testing sets at a 7:3 ratio. The training set was further divided into training and validation sets at an 8:2 ratio. The testing set was used to evaluate the final performance of our model.

Evaluation metrics

In the experiments, we selected the number of features, F1-Macro, accuracy, model prediction speed, F1-Weighted, and AUC as evaluation metrics. The number of features is used to measure whether the feature selection algorithm can extract more representative features, reduce redundancy, and enhance the model's generalization ability. It also provides a basis for constructing datasets for fine-tuning large language models in subsequent tasks. F1-Macro is used to evaluate the overall performance of the model after data augmentation, especially in cases where the class distribution is imbalanced. This metric can more comprehensively reflect the model's performance. Accuracy is used to measure the model's prediction capability and to assess the model's performance when using only the selected features. Additionally, accuracy can reflect the quality of data augmentation; if the accuracy decreases after data augmentation, it indicates that the generated data may contain noise and is of lower quality. Given the complexity and large volume of network traffic in the field of IoT

NF-CSE-CIC-IDS2018-v2		NF-ToN-IoT-v2		CIC-ToN-IoT		NF-UNSW-NB15-v2		NF-BoT-IoT-v2	
Class	**Count**	**Class**	**Count**	**Class**	**Count**	**Class**	**Count**	**Class**	**Count**
DDOS attack-HOIC	1,080,858	Benign	6,099,469	Benign	2,515,236	Analysis	2299	Benign	135,037
DoS attacks-Hulk	432,648	Scanning	3,781,419	Backdoor	27,145	Backdoor	2169	DDoS	18,331,847
DDoS attacks-LOIC-HTTP	307,300	xss	2,455,020	dos	145	Benign	2,295,222	DoS	16,673,183
Infilteration	116,361	ddos	2,026,234	ddos	202	DoS	5,794	Theft	2,431
SSH-Bruteforce	94,979	Password	1,153,323	Injection	277,696	Exploits	31,551	Reconnaissance	2,620,999
Bot	143,097	dos	712,609	mitm	517	Fuzzers	22,310		
DoS attacks-GoldenEye	27,723	Injection	684,465	Password	340,208	Generic	16,560		
FTP-BruteForce	25,933	Backdoor	16,809	Ransomware	5098	Reconnaissance	12,779		
DoS attacks-SlowHTTPTest	14,116	mitm	7723	Scanning	36,205	Worms	164		
DoS attacks-Slowloris	9512	Ransomware	3425	xss	2,149,308	Shellcode	1427		
Brute Force -Web	2143								
DDOS attack-LOIC-UDP	2112								
Brute Force -XSS	927								
SQL Injection	432								
Benign	16,635,567								
Total	18,893,708	Total	16,940,496	Total	5,351,760	Total	2,390,275	Total	37,763,497

Table 1. Distribution of attack types in five datasets.

intrusion detection, the detection speed of the model is crucial. We selected model prediction speed as a metric to measure the model's inference efficiency on the test set, thereby ensuring its applicability in real IoT scenarios.

F1-Weighted is a weighted average of the F1 scores for each class, where the weights are based on the number of samples in each class. This metric is particularly useful for evaluating overall performance on imbalanced datasets. In the formula below, TP_i, FP_i, and FN_i represent the true positives, false positives, and false negatives for class i, respectively, and S_i is the number of instances of class i:

$$\text{F1-Weighted} = \frac{\sum_{i=1}^{n} \frac{2 \cdot TP_i}{2 \cdot TP_i + FP_i + FN_i} \cdot S_i}{\sum_{i=1}^{n} S_i} \tag{14}$$

F1-Macro calculates the F1 score for each class separately and then takes the arithmetic mean of these scores. This metric is well-suited for evaluating the effectiveness of data augmentation algorithms. In the formula below, N is the total number of classes:

$$\text{F1-Macro} = \frac{1}{N} \sum_{i=1}^{N} \frac{2 \cdot P_i \cdot R_i}{P_i + R_i} \tag{15}$$

The Area Under the Curve (AUC) provides a comprehensive method for evaluating model performance, particularly suitable for imbalanced datasets and situations requiring a balance between different types of errors. Consequently, we have presented the AUC metric performance for five datasets in this study. As AUC is typically applied to binary classification, we employed the OvR strategy to convert multi-class classification into multiple binary classification tasks for calculation. Therefore, the AUC metric for each class can be calculated using Equation 16, where $\text{TPR}_i = \frac{TP_i}{TP_i + FN_i}$ and $\text{FPR}_i = \frac{FP_i}{FP_i + TN_i}$.

$$\text{AUC}_i = \int_0^1 \text{TPR}_i(\text{FPR}_i) d(\text{FPR}_i) \tag{16}$$

Implementation and experimental environment

1. For the MCP feature selection algorithm, the experimental setup includes the following: Python 3.10, AMD Ryzen 5 7600 CPU, RTX 3090 8GB GPU, 32 GB RAM, Fast-mRMR 1.0, cmaes 0.10, scikit-learn 1.3.2, and LightGBM 4.3. For the experiments, we set the number of iterations for the feature selection algorithm to 3, the population size and maximum iterations for CMA-ES to 20, and we use the default parameters for LightGBM. To ensure reproducibility, we set the random seed to 42. The experimental process employs F1-Macro as the fitness metric for the CMA-ES algorithm, aiming to minimize both the number of features and the performance loss.

Figure 4 illustrates the feature selection process of the MCP algorithm across five datasets. For NF-CSE-CIC-IDS2018-v2, the validation set scores for the first and last iterations were 0.98862 and 0.988544, respectively. NF-UNSW-NB15-v2 yielded scores of 0.997151 and 0.997144 for the first and last iterations. NF-ToN-IoT-v2

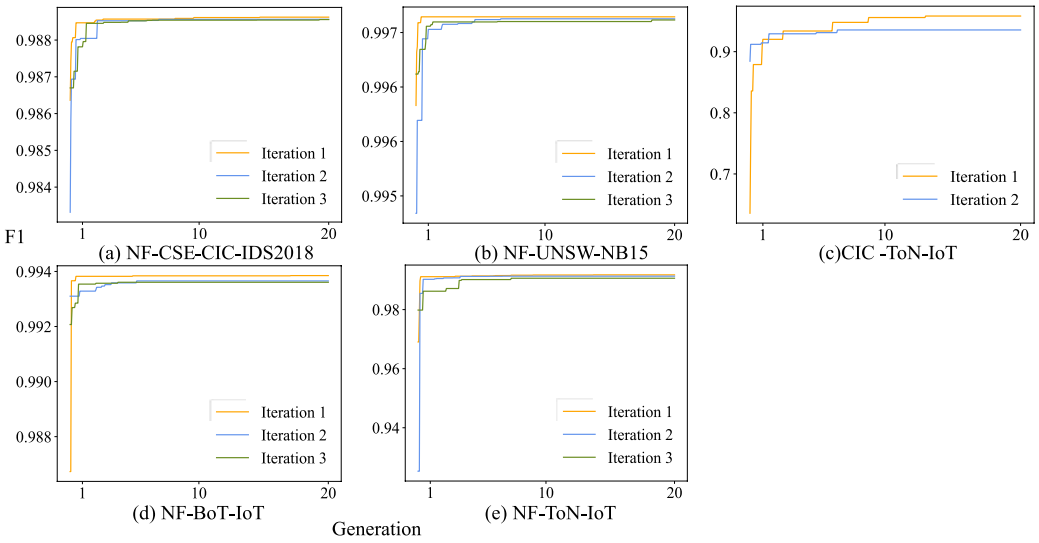

Fig. 4. The MCP algorithm was applied to five datasets. The search process terminates early when no collinear features are detected during these iterations. For example, in the case of the NF-BoT-IoT-v2 dataset, the iteration process stopped after two iterations due to the absence of collinear features.

Dataset	Features
NF-CSE-CIC-IDS2018-v2	CLIENT_TCP_FLAGS, DNS_TTL_ANSWER, MAX_TTL, L4_DST_PORT, SHORTEST_FLOW_PKT, SRC_TO_DST_AVG_THROUGHPUT, IN_PKTS, SERVER_TCP_FLAGS, TCP_WIN_MAX_IN, DST_TO_SRC_AVG_THROUGHPUT
NF-ToN-IoT-v2	TCP_WIN_MAX_IN, TCP_WIN_MAX_OUT, FLOW_DURATION_MILLISECONDS, L4_DST_PORT, DST_TO_SRC_AVG_THROUGHPUT, L7_PROTO, IN_BYTES
NF-UNSW-NB15-v2	MAX_TTL, ICMP_TYPE, L4_DST_PORT, TCP_FLAGS, MAX_IP_PKT_LEN, L7_PROTO, PROTOCOL, SRC_TO_DST_AVG_THROUGHPUT, OUT_BYTES
NF-BoT-IoT-v2	L4_SRC_PORT, DURATION_OUT, RETRANSMITTED_OUT_PKTS, CLIENT_TCP_FLAGS, NUM_PKTS_256_TO_512_BYTES, TCP_FLAGS, NUM_PKTS_UP_TO_128_BYTES, NUM_PKTS_1024_TO_1514_BYTES, SRC_TO_DST_SECOND_BYTES
CIC-ToN-IoT	Src Port, Dst Port, Init Fwd Win Byts, Init Bwd Win Byts, Idle Min, Bwd Pkt Len Mean, Pkt Len Min, Flow IAT Mean, Idle Max, Active Max, Bwd IAT Std, Active Std, Fwd Act Data Pkts

Table 2. Feature subsets selected by MCP algorithm in five datasets.

showed scores of 0.991761 and 0.990622, while NF-BoT-IoT-v2 produced scores of 0.993850 and 0.993607. For these four datasets, the performance difference between the first and last iterations after three iterations of adaptive threshold-based collinear feature removal was less than 0.001. The CIC-ToN-IoT dataset eliminated all collinear features above the threshold after two iterations, with scores of 0.958294 and 0.935535, resulting in a performance difference of 0.02. This demonstrates that the iterative search strategy for adaptively adjusting the PCC threshold can better remove redundant features while causing minimal performance loss. The approach effectively balances feature reduction and model performance across different datasets. Table 2 demonstrates that the MCP algorithm identified 9 features in NF-CSE-CIC-IDS2018-v2, NF-UNSW-NB15-v2, and NF-BoT-IoT-v2 datasets, 7 features in NF-ToN-IoT-v2, and 13 features in CIC-ToN-IoT. Compared to the original feature sets, our algorithm reduced the number of features by over 80%.

2. For the LLMs based-data augmentation algorithm, we fine-tune our datasets using the GPT-2 large language model[45], employing the features detailed in Table 2. The fine-tuning environment utilizes Python 3.10, be-great 0.0.7[46], Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz, V100-32GB GPU, and scikit-learn 1.3.2. During sampling, a temperature parameter T of 0.7 is applied, with a batch size of 64. We employ AdamW as the default optimizer with an initial learning rate of 5e-5.

Based on the number of samples in minority attack classes within each dataset, we set different fine-tuning epochs. For NF-CSE-IDS2018-v2, NF-UNSW-NB15-v2, and CIC-ToN-IoT datasets, fine-tuning continues for 300 epochs, while NF-BoT-IoT-v2 and NF-ToN-IoT-v2 are fine-tuned for 20 epochs. We generate 10,000 samples for each minority class within the dataset, and then remove duplicate samples from the generated samples. During the testing process, we utilize the same hardware environment as in (1) while employing the FL improved-LightGBM classifier.

Method	NF-CSE-CIC-IDS2018-v2			NF-ToN-IoT-v2			NF-UNSW-NB15-v2			NF-BoT-IoT-v2			CIC-ToN-IoT		
	num	F1	Acc	num	F1	Acc	num	F1	Acc	num	F1	Acc	num	F1	Acc
Fast-mRMR	33	0.995	0.995	35	0.991	0.991	36	0.997	0.997	23	0.998	0.998	43	0.993	0.993
CMA-ES	21	0.995	0.995	26	0.991	0.991	22	0.997	0.997	17	0.998	0.998	36	0.993	0.993
GA	23	0.995	0.995	26	0.991	0.991	25	0.997	0.997	22	**1.0**	**1.0**	41	0.993	0.993
PSO	20	0.995	0.995	25	0.991	0.991	19	0.997	0.997	24	0.928	0.872	39	0.993	0.993
Mohy et al.[25]	–	–	–	–	–	–	24	0.992	0.993	–	–	–	–	–	–
Leevy et al.[24]	9	0.990	0.990	9	0.992	0.992	9	0.983	0.983	9	0.989	0.987	–	–	–
Sarhan et al.[9]	8	0.840	0.955	8	**1.0**	**0.994**	**8**	0.850	0.985	–	–	–	–	–	–
MCP	**9**	**0.995**	**0.995**	**7**	0.990	0.990	9	**0.997**	**0.997**	**9**	0.997	0.997	**13**	**0.993**	**0.993**

Table 3. The results of binary classification using features selected by the MCP algorithm.

Method	NF-CSE-CIC-IDS2018-v2			NF-ToN-IoT-v2			NF-UNSW-NB15-v2			NF-BoT-IoT-v2			CIC-ToN-IoT		
	num	F1	Acc	num	F1	Acc	num	F1	Acc	num	F1	Acc	num	F1	Acc
Fast-mRMR	33	0.968	0.972	35	0.943	0.935	36	0.983	0.984	23	0.925	0.917	43	0.829	0.861
CMA-ES	21	0.753	0.630	26	0.947	0.945	22	0.975	0.976	17	0.976	0.976	36	**0.831**	0.853
GA	23	0.918	0.922	26	0.945	0.944	25	0.971	0.971	22	0.978	0.978	41	0.823	0.856
PSO	20	0.811	0.851	25	0.946	0.947	19	0.971	0.973	24	0.974	0.974	39	0.818	0.850
Leevy et al.[24]	9	0.960	0.975	9	0.620	0.705	9	0.960	0.972	9	0.830	0.837	9	0.820	0.870
MCP	9	0.975	0.977	7	0.949	0.949	9	0.978	0.977	9	0.967	0.968	13	0.830	0.849
MCP(FL)	**9**	**0.984**	**0.984**	**7**	**0.956**	**0.956**	**9**	**0.992**	**0.992**	**9**	**0.988**	**0.988**	13	0.825	**0.873**

Table 4. The results of multi-classification using features selected by the MCP algorithm.

3. For the FL improved-LightGBM classifier, the parameter α_t in the FL function is set to 0.75, and γ is set to 2.0.
4. The environment used in the comparative experiments is identical to the one employed in (1) and (3).

Experimental analysis of MCP algorithm

In this section, we evaluate the performance of the MCP feature selection algorithm across five datasets. We compare it with recently published algorithms and classical methods, as summarized in Table 3 and Table 4. The GA[10] and PSO[11] algorithm are implemented using MEALPY[47], representing heuristic algorithms. Sarhan et al.[9] uses the chi-square test, information gain, and PCC for feature selection. Leevy et al.[24] employs information gain, while Mohy et al.[25] combines Isolation Forest with the PCC for feature selection.

1. For binary classification (Table 3), the features selected by the MCP algorithm demonstrate exceptional performance, achieving F1-weighted scores and accuracy rates exceeding 0.990 across all datasets. The features selected by the GA algorithm on the NF-BoT-IoT-v2 dataset yield accuracy and F1-weighted scores 0.3% higher than those selected by our algorithm; however, our algorithm selects 13 fewer features. The algorithm used in Sarhan et al.[9] selects features that achieve 1% higher accuracy and 0.4% higher F1-weighted score on NF-ToN-IoT-v2 compared to our algorithm. Nevertheless, Sarhan et al.[9] selects one more feature than our algorithm and shows clear disadvantages on the remaining datasets. The MCP algorithm demonstrates significant advantages over the algorithms used in Leevy et al.[24] and Mohy et al.[25], both in terms of the number and quality of selected features. The features selected by Fast-mRMR and CMA-ES algorithms also achieve high F1-weighted scores and accuracy rates, indicating their strong performance in binary classification tasks. However, they select considerably more features than the MCP algorithm. In summary, for binary classification tasks, the features selected by our algorithm exhibit more stable performance and generalization capability compared to other algorithms.
2. For multi-class classification (Table 4), the MCP algorithm demonstrates excellent performance, maintaining high F1-Weighted scores and accuracy while selecting fewer features (7-13) across all datasets. On the NF-CSE-CIC-IDS2018-v2 dataset, the MCP algorithm achieves an F1-Weighted score of 0.975 and accuracy of 0.977 using 9 features, outperforming Fast-mRMR (33 features, F1-Weighted 0.968, accuracy 0.972) and other algorithms. For NF-ToN-IoT-v2 dataset, the MCP algorithm attains an F1-Weighted score and accuracy of 0.949 with only 7 features, comparable to other algorithms selecting 25-26 features (F1-Weighted and accuracy between 0.945-0.947), and significantly superior to the algorithm in Leevy et al.[24] (9 features, F1-Weighted 0.620, accuracy 0.705). On NF-UNSW-NB15-v2 dataset, the MCP algorithm obtains the highest F1-Weighted score of 0.978 and a near-highest accuracy of 0.977 with 9 features, while other algorithms achieve similar performance (F1-Weighted 0.971-0.983, accuracy 0.971-0.984) with 19-36 features. For the NF-BoT-IoT-v2 dataset, the MCP algorithm yields an F1-Weighted score of 0.967 and accuracy of 0.968

using 9 features, slightly lower than CMA-ES (17 features, F1-Weighted and accuracy both 0.976) and GA (22 features, F1-Weighted and accuracy both 0.978), but with fewer features. It outperforms Fast-mRMR (23 features, F1-Weighted 0.925, accuracy 0.917). On CIC-ToN-IoT dataset, the MCP algorithm selects 13 features, achieving an F1-Weighted score of 0.830 and accuracy of 0.849, comparable to other algorithms using 36-43 features (F1-Weighted 0.818-0.831, accuracy 0.850-0.861). With the FL function, the MCP algorithm's performance further improves on most datasets, surpassing other algorithms except for a marginally lower F1-Weighted score (0.5% difference) compared to GA algorithm on CIC-ToN-IoT dataset. In summary, the MCP algorithm exhibits robust feature selection capabilities, achieving or exceeding the performance of other algorithms with fewer features.

As shown in Fig. 5, to further analyze the performance of the MCP feature selection algorithm, we compared model performance when using all features versus selected features obtained through the feature selection algorithm (the experimental process uniformly employed an FL-function-improved LightGBM as the classifier). Experimental results demonstrate that feature selection effectively reduces computational complexity while maintaining classification performance. Across all datasets, the accuracy after feature selection remains nearly identical to results without feature selection. For instance, both the NF-UNSW-NB15-v2 and NF-CSE-CIC-IDS2018-v2 datasets achieved accuracies of 0.992 and 0.984, respectively, indicating that feature selection does not compromise overall classification capability. Meanwhile, the F1-Weighted metric shows minimal variation across different datasets, further verifying that feature selection preserves critical information, enabling classification models to maintain stable decision-making. Notably, the F1-Macro metric demonstrates significant improvement on certain datasets. For example, the NF-ToN-IoT-v2 dataset shows an F1-Macro increase to 0.761 after feature selection, compared to only 0.732 without feature selection. This suggests that the feature selection method enhances classification performance for minority classes, thereby improving model generalization. These findings reveal that in network intrusion detection and IoT security scenarios, feature selection not only reduces computational overhead but can also enhance recognition capability for minority classes under certain conditions, enabling more stable model performance in imbalanced data scenarios (leveraging the FL-function-enhanced LightGBM classifier).

Experimental analysis of LLMs based-data augmentation algorithms

This section evaluates the data augmentation algorithm based on LLMs within our framework, focusing on data augmentation requirements for multi-class classification. We compare our approach with baseline methods, including CTGAN[30], SMOTE[15], ADASYN[28,29], Random Oversampling[14], and Random Undersampling.

As shown in Table 5, on the NF-CSE-CIC-IDS2018-v2 dataset, our method achieved an F1-Macro of 0.794 and an accuracy of 0.995, both being the highest values, significantly surpassing SMOTE (F1-Macro 0.697) and the original data without augmentation (F1-Macro 0.678). CTGAN performed relatively well on this dataset (F1-Macro 0.780) but was still slightly lower than our method, indicating that our approach has stronger adaptability in data generation and feature learning. For DDoS and DoS attacks (such as DoS attacks-Hulk, DDoS attacks-LOIC-HTTP, and DoS attacks-GoldenEye), our algorithm achieved a classification performance of 1.0, comparable to CTGAN and SMOTE, but significantly higher than the original data without augmentation (e.g., the detection value for DoS attacks-GoldenEye using the original data was only 0.994). However, for more challenging attack types such as SQL injection and cross-site scripting (XSS), our algorithm showed an improvement over the original data (SQL injection increased from 0.028 to 0.079) but still performed lower than SMOTE (0.002). This suggests that under extremely imbalanced data conditions, our method still has room for improvement.

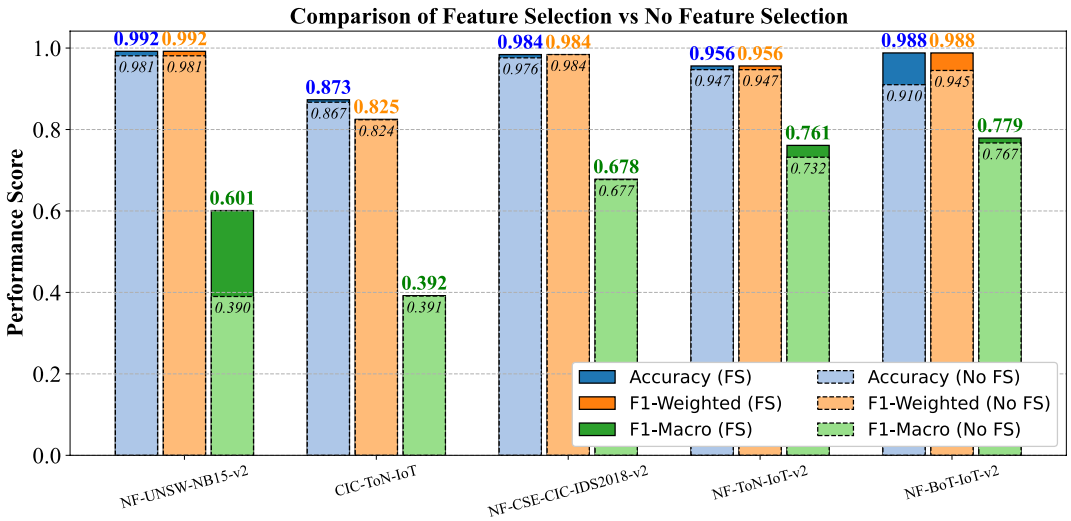

Fig. 5. Comparison of feature selection vs no feature selection.

	Our	CTGAN	SMOTE	ADASYN	Oversample	Undersample	Original
DDOS attack-HOIC	0.998	0.998	0.705	–	0.625	0.543	0.998
DoS attacks-Hulk	1.0	1.0	1.0	–	1.0	0.996	0.995
DDoS attacks-LOIC-HTTP	1.0	1.0	1.0	–	1.0	0.979	1.0
Infilteration	0.471	0.467	0.03	–	0.03	0.031	0.476
SSH-Bruteforce	1.0	1.0	1.0	–	1.0	0.877	1.0
Bot	1.0	1.0	1.0	–	1.0	0.991	1.0
DoS attacks-GoldenEye	1.0	1.0	1.0	–	1.0	0.963	0.994
FTP-BruteForce	1.0	1.0	1.0	–	1.0	0.919	1.0
DoS attacks-SlowHTTPTest	1.0	1.0	1.0	–	1.0	0.930	1.0
DoS attacks-Slowloris	1.0	1.0	1.0	–	1.0	0.759	0.667
Brute Force -Web	0.205	0.247	0.01	–	0.009	0.007	0.028
DDOS attack-LOIC-UDP	0.993	0.987	0.982	–	0.992	0.536	0.0
Brute Force -XSS	0.163	0.0	0.003	–	0.003	0.002	0.0
SQL Injection	0.079	0.0	0.002	–	0.002	0.001	0.028
Benign	0.997	0.996	0.725	–	0.708	0.802	0.991
F1-Macro	**0.794**	0.780	0.697	–	0.691	0.622	0.678
Accuracy	**0.995**	0.992	0.591	–	0.568	0.668	0.984

Table 5. The performance of LLMs based-data augmentation on the NF-CSE-CIC-IDS2018-v2 dataset.

	Our	CTGAN	SMOTE	ADASYN	Oversample	Undersample	Original
Benign	0.955	0.956	0.934	0.885	0.934	0.836	0.952
DDoS	0.993	0.993	0.992	0.992	0.992	0.990	0.993
DoS	0.987	0.987	0.987	0.828	0.987	0.976	0.987
Theft	0.966	0.966	0.965	0.550	0.965	0.918	0.965
Reconnaissance	0.729	0.749	0.696	0.624	0.696	0.349	0.0
F1-Macro	0.926	**0.930**	0.915	0.776	0.915	0.814	0.779
Accuracy	**0.988**	**0.988**	**0.988**	0.866	**0.988**	0.977	**0.988**

Table 6. The performance of LLMs based-data augmentation on the NF-BoT-IoT-v2 dataset.

	Our	CTGAN	SMOTE	ADASYN	Oversample	Undersample	Original
Benign	0.983	0.975	0.982	0.975	0.984	0.968	0.982
Scanning	0.987	0.986	0.990	0.982	0.989	0.976	0.986
XSS	0.944	0.945	0.946	0.878	0.946	0.938	0.944
DDoS	0.970	0.969	0.972	0.941	0.972	0.960	0.967
Password	0.915	0.914	0.921	0.878	0.922	0.907	0.915
DoS	0.903	0.903	0.904	0.482	0.903	0.901	0.902
Injection	0.822	0.820	0.830	0.775	0.835	0.785	0.822
Backdoor	0.995	0.386	0.986	0.977	0.984	0.972	0.992
MITM	0.117	0.035	0.088	0.032	0.102	0.073	0.064
Ransomware	0.903	0.932	0.881	0.870	0.682	0.565	0.033
F1-Macro	**0.854**	0.786	0.850	0.779	0.832	0.804	0.761
Accuracy	**0.958**	0.953	0.957	0.902	0.957	0.945	0.956

Table 7. The performance of LLMs based-data augmentation on the NF-ToN-IoT-v2 dataset.

As shown in Table 6, on the NF-BoT-IoT-v2 dataset, our method outperformed the original data without augmentation across all categories and surpassed CTGAN, SMOTE, and ADASYN in multiple categories, ultimately achieving an F1-Macro of 0.926 and an accuracy of 0.988. In the detection of DDoS and DoS attacks, our method, CTGAN, and SMOTE performed very similarly (with F1 scores close to 0.99), indicating that these methods can achieve good results for high-frequency attack categories. However, in the detection of Reconnaissance attacks, our method (0.729) outperformed the original data and Undersampling (0.349) but was slightly lower than CTGAN (0.749). This may be related to CTGAN's better balance when synthesizing minority

	Our	CTGAN	SMOTE	ADASYN	Oversample	Undersample	Original
Analysis	0.118	0.03	0.149	0.136	0.113	0.145	0.060
Backdoor	0.404	0.30	0.258	0.252	0.237	0.225	0.314
Benign	0.998	0.998	0.998	0.998	0.997	0.995	0.998
DoS	0.367	0.273	0.378	0.365	0.404	0.218	0.358
Exploits	0.875	0.869	0.783	0.770	0.812	0.638	0.871
Fuzzers	0.895	0.893	0.856	0.865	0.876	0.827	0.889
Generic	0.806	0.788	0.729	0.545	0.736	0.552	0.796
Reconnaissance	0.904	0.887	0.885	0.793	0.880	0.876	0.889
Worms	0.928	0.928	0.870	0.876	0.914	0.350	0.927
Shellcode	0.649	0.420	0.398	0.437	0.217	0.229	0.0
F1-Macro	**0.701**	0.639	0.630	0.603	0.618	0.505	0.610
Accuracy	**0.992**	0.992	0.989	0.987	0.987	0.979	**0.992**

Table 8. The performance of LLMs based-data augmentation on the NF-UNSW-NB15-v2 dataset.

	Our	CTGAN	SMOTE	ADASYN	Oversample	Undersample	Original
Benign	0.993	0.993	0.993	0.993	0.993	0.975	0.993
XSS	0.863	0.863	0.740	0.727	0.718	0.672	0.863
Password	0.069	0.081	0.405	0.398	0.404	0.280	0.059
Injection	0.047	0.035	0.407	0.398	0.409	0.216	0.035
Scanning	0.003	0.0	0.222	0.211	0.216	0.126	0.0
Backdoor	0.994	0.995	0.994	0.995	0.996	0.928	0.996
Ransomware	0.961	0.965	0.414	0.916	0.931	0.195	0.969
MITM	0.580	0.555	0.380	0.396	0.463	0.031	0.0
DDoS	0.233	0.051	0.056	0.035	0.054	0.005	0.0
DoS	0.315	0.08	0.059	0.074	0.031	0.065	0.0
F1-Macro	**0.506**	0.462	0.467	**0.514**	**0.522**	0.349	0.392
Accuracy	**0.873**	**0.873**	0.789	0.782	0.778	0.716	0.873

Table 9. The performance of LLMs based-data augmentation on the CIC-ToN-IoT dataset.

class data. However, CTGAN's performance is inconsistent across multiple datasets, whereas our algorithm still demonstrates superior overall generalization capability compared to other methods.

As shown in Table 7, on the NF-ToN-IoT-v2 dataset, our method demonstrated significantly better detection performance in multiple key attack categories (such as DDoS, XSS, and password attacks) compared to CTGAN and SMOTE, ultimately achieving an F1-Macro of 0.854 and an accuracy of 0.958. CTGAN achieved an F1-Macro of only 0.786 on this dataset, while SMOTE reached 0.850, both lower than our algorithm. In Backdoor attack detection, our method achieved 0.995, significantly outperforming CTGAN (0.386), indicating that our method can still learn valuable feature information under extreme data imbalance conditions. Additionally, for DoS and Injection attack detection, our method achieved F1 scores of 0.903 and 0.822, respectively, surpassing all other comparison methods, demonstrating its high detection capability across different attack patterns. However, there is still room for optimization in detecting MITM and Ransomware attacks. For instance, in MITM attack detection, our method achieved an F1 score of only 0.117, slightly higher than CTGAN (0.035) and SMOTE (0.088), but still relatively low. This suggests that the characteristics of such attacks are more difficult to capture, and future work should focus on optimizing data augmentation strategies.

As shown in Table 8, the NF-UNSW-NB15-v2 dataset contains a large number of attack types, leading to relatively lower performance for most data augmentation methods. However, our method still achieved an F1-Macro of 0.701 and an accuracy of 0.992, ranking the highest among all methods. CTGAN (0.639) and SMOTE (0.630) had significantly lower F1-Macro scores than our algorithm, indicating their poorer adaptability to this dataset. Our method outperformed other approaches in attack categories such as Exploit, Fuzzers, Generic, and Reconnaissance. For example, in Exploit detection, our method achieved 0.875, showing a clear improvement over CTGAN (0.869) and SMOTE (0.783). Additionally, in Reconnaissance attack detection, our method achieved an F1 score of 0.904, significantly surpassing SMOTE (0.885) and CTGAN (0.887). However, there is still room for improvement in detecting challenging categories such as Analysis and Backdoor. For instance, in Backdoor detection, our method achieved 0.404, which, while better than CTGAN (0.300) and SMOTE (0.258), remains relatively low. This suggests that the generalization ability of our method for extremely low-frequency attack categories still needs further enhancement.

As shown in Table 9, on the CIC-ToN-IoT dataset, our method achieved an overall F1-Macro of 0.506 and an accuracy of 0.873, significantly outperforming the original data without augmentation (0.392). Compared to

CTGAN (0.462) and SMOTE (0.467), our method achieved better detection performance across multiple attack categories, particularly in XSS (0.863) and Ransomware (0.961) detection, where it achieved the best results. However, for low-frequency attack categories such as password attacks and SQL injection, SMOTE (0.405 and 0.407) performed slightly better than our method (0.069 and 0.047). This may be because SMOTE can synthesize minority class samples more effectively in these categories. Nevertheless, in DDoS and DoS attack detection, our method continued to surpass other approaches, maintaining high stability.

Overall, our method demonstrates higher accuracy and stability across multiple datasets, particularly excelling in common attack categories such as DDoS, DoS, and Backdoor attacks, where it exhibits stronger adaptability and, in most cases, outperforms CTGAN, SMOTE, and other data augmentation methods. Furthermore, our method achieves the highest overall F1-Macro and accuracy on most datasets, indicating superior generalization capability. In contrast, CTGAN performs well in certain specific categories (e.g., SQL injection and low-frequency attacks), while SMOTE occasionally has advantages in some low-frequency categories but still falls short of our algorithm overall. Therefore, our method demonstrates superior comprehensive performance in terms of balance, generalization ability, and complex attack detection. Future improvements could incorporate feature selection and optimized data augmentation strategies to further enhance detection capability for low-frequency attack categories.

Comparative experiments of FSLLM

In the preceding sections, we conducted comparative analyses of the feature selection and data augmentation modules within the FSLLM framework against other algorithms. To further validate the performance and generalization capability of the FSLLM framework, this section presents a comparative analysis with recent state-of-the-art IoT intrusion detection algorithms. The selection of these methods is based on their representativeness and advancements in the field, ensuring a fair and comprehensive comparison. The comparison includes:

Nguyen et al.[22]: A self-supervised graph neural network algorithm designed for IoT intrusion detection. This method is chosen due to its ability to capture complex graph-structured relationships in IoT networks, representing an advanced approach in self-supervised learning for cybersecurity.

Li et al.[32]: An approach integrating BERT and CGAN to enhance intrusion detection. This method is included as it represents a cutting-edge combination of pre-trained language models and generative techniques, demonstrating high adaptability in learning attack patterns.

Termos et al.[7]: A graph deep learning approach based on centrality measures. This method is chosen as it leverages graph theory principles to enhance anomaly detection, representing a novel perspective in network security.

Wang et al.[23]: A spatio-temporal graph attention network that considers node states for intrusion detection. This approach is selected due to its effectiveness in modeling temporal and spatial dependencies, making it a strong baseline for IoT security applications.

Sarhan et al.[8]: A deep learning-based anomaly detection algorithm. This method is widely recognized in the field for its ability to detect novel and sophisticated attack patterns using unsupervised learning.

All selected algorithms were published between 2023 and 2024, representing the latest trends in IoT intrusion detection research. Additionally, these methods utilize the same experimental datasets as our study, ensuring a direct and fair comparison. However, since some of the selected methods did not provide open-source code, we report their results based on the performance metrics presented in their original papers. These benchmark methods serve as a unified baseline, incorporating both traditional deep learning and advanced graph-based approaches, ensuring a comprehensive evaluation of FSLLM's performance. By comparing FSLLM against these representative and state-of-the-art algorithms, we aim to demonstrate its superiority in terms of detection accuracy, efficiency, and generalization capability.

Method	Dataset	Features	Binary cassification		Multi-classification	
			F1-weighted	Accuracy	F1-weighted	Accuracy
Nguyen et al.[22]	NF-CSE-CIC- IDS2018-v2	All	0.995	0.995	0.992	-
	NF-UNSW-NB15-v2	All	0.997	0.997	0.990	-
Li et al.[32]	NF-ToN-IoT-v2	All	-	-	0.988	0.988
	NF-UNSW-NB15-v2	All	-	-	0.890	0.874
Sarhan et al.[8]	NF-CSE-CIC- IDS2018-v2	All	0.889	0.891	-	-
	NF-UNSW-NB15-v2	All	0.983	0.982	-	-
Wang et al.[23]	NF-ToN-IoT-v2	All	0.979	0.966	0.914	-
	NF-BoT-IoT-v2	All	0.987	0.979	0.926	-
Termos et al.[7]	CIC-ToN-IoT	All	0.986	0.989	0.806	0.857
FSLLM	NF-CSE-CIC- IDS2018-v2	9	**0.995**	**0.995**	0.988	0.988
	NF-ToN-IoT-v2	7	**0.990**	**0.990**	0.958	0.958
	NF-UNSW-NB15-v2	9	**0.997**	**0.997**	**0.992**	**0.992**
	NF-BoT-IoT-v2	9	**0.997**	**0.997**	**0.988**	**0.988**
	CIC-ToN-IoT	13	**0.993**	**0.993**	**0.825**	**0.873**

Table 10. Comparative experimental results of FSLLM gramework.

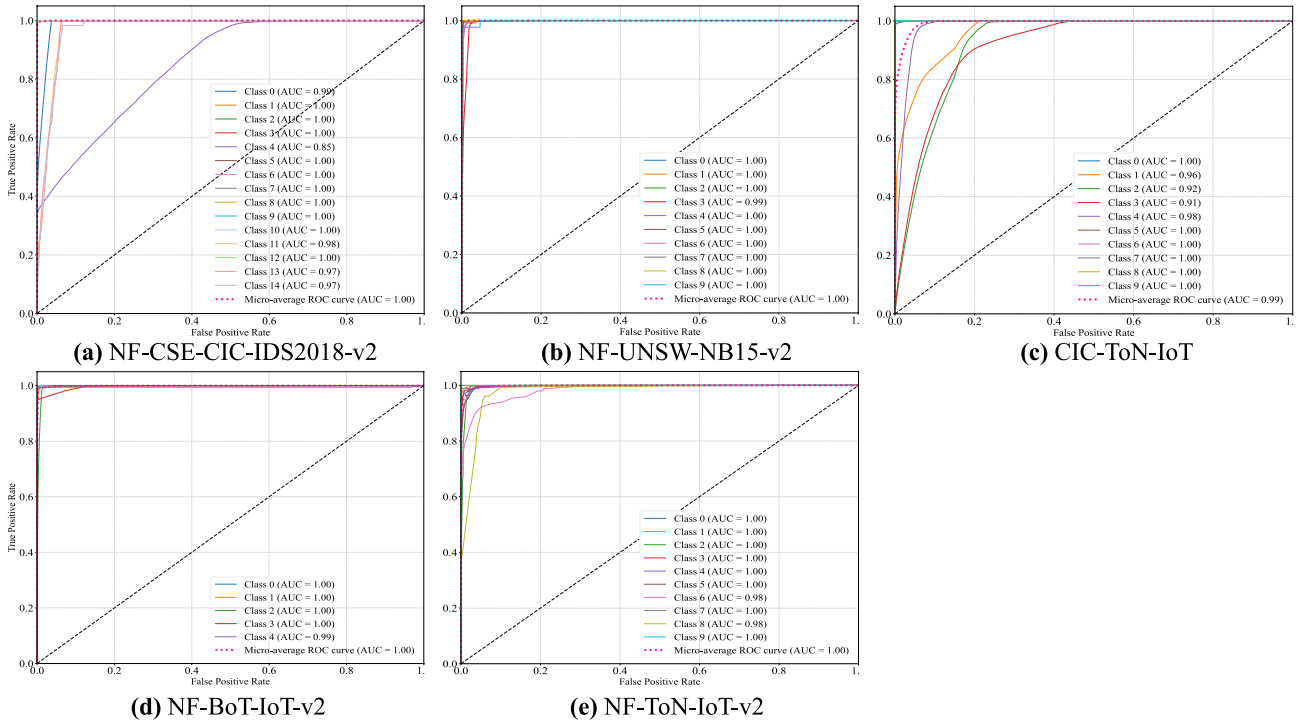

Fig. 6. AUC metric performance across five datasets.

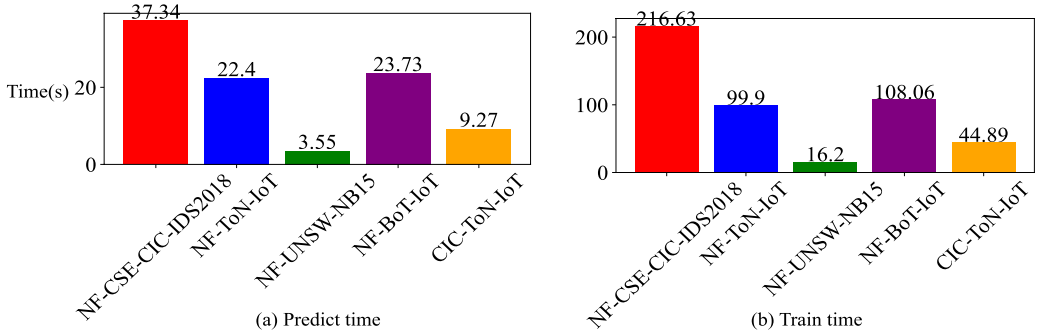

Fig. 7. Training and prediction times of FL improved-LightGBM across different datasets.

Experimental results (Table 10) demonstrate that our proposed method exhibits exceptional performance across multiple IoT intrusion detection datasets, surpassing or matching state-of-the-art techniques in most cases. For binary classification tasks, our method achieves the highest or equal-to-highest F1-Weighted scores and accuracies on the NF-CSE-CIC-IDS2018-v2, NF-ToN-IoT-v2, NF-UNSW-NB15-v2, NF-BoT-IoT-v2, and CIC-ToN-IoT datasets, with scores of 0.995, 0.990, 0.997, 0.997, and 0.993 respectively. These results are superior to or match the best performances reported in previous studies such as Nguyen et al.[22], Wang et al.[23], and Sarhan et al.[8].

In the more challenging multi-class classification tasks, our method also excels. On the NF-UNSW-NB15-v2 dataset, our method attains an F1-Weighted score and accuracy of 0.992, significantly outperforming the 0.890 and 0.874 reported in Li et al.[32]. For the NF-BoT-IoT-v2 and CIC-ToN-IoT datasets, our method achieves F1-Weighted scores of 0.988 and 0.825, respectively, surpassing the results in Wang et al.[23] and Termos et al.[7]. Although our F1-Weighted score of 0.958 on the NF-ToN-IoT-v2 dataset is slightly lower than the 0.988 reported in Li et al.[32], it still outperforms other comparative methods.

As illustrated in Fig. 6, we present the AUC metric performance across five datasets. The results demonstrate that the majority of classes in all five datasets exhibit AUC values exceeding 0.98, with only one class in the NF-CSE-CIC-IDS2018-v2 dataset achieving a score of 0.85. Notably, our method utilizes only a small number of selected features (7-13), while most comparative methods use all available features. These characteristic highlights both the efficiency of our approach and its superior feature selection capability and generalization ability. These results strongly support the effectiveness of our pro-posed method and its applicability across various IoT security scenarios.

Efficiency analysis of FSLLM

We test the training and prediction times of LightGBM improved with the FL function on various datasets under the environment described in "Implementation and experimental environment" section. The specific results are shown in Fig. 7. Our algorithm exhibits an average training time of 97.94 s and an average prediction time of 19.66 seconds across different datasets, demonstrating its exceptional computational efficiency. Our algorithm completes training and prediction within a short period across datasets of varying sizes, effectively adapting to the complex environment of IoT intrusion detection.

Conclusions

This study investigates the interplay between feature redundancy and class imbalance in the context of IoT intrusion detection. We propose the FSLLM framework, which utilizes feature selection algorithms to mitigate feature redundancy and generate fine-tuning datasets for LLMs using exclusively selected representative features. These carefully selected non-redundant and highly representative features facilitate the generation of high-quality samples while simultaneously reducing computational costs associated with fine-tuning. The efficacy of our proposed method is underpinned by several key innovations: (1) The MCP feature selection algorithm, which integrates mRMR and PCC-improved CMA-ES algorithm, considers both mutual information and collinearity among features, thereby effectively eliminating redundancy. (2) The integration of MCP feature selection with LLMs, which results in a reduction of fine-tuning dataset size and an enhancement of overall data quality. (3) The utilization of FL-improved LightGBM as a classifier, which significantly enhances intrusion detection performance. The FSLLM framework was evaluated using five distinct datasets: NF-CSE-CIC-IDS2018-v2, NF-ToN-IoT-v2, NF-UNSW-NB15-v2, NF-BoT-IoT-v2, and CIC-ToN-IoT. The experimental results demonstrate that FSLLM substantially mitigates feature redundancy, eliminating over 80% of redundant features while concurrently maintaining or enhancing detection accuracy in comparison to state-of-the-art algorithms. Notwithstanding the promising results, this study acknowledges certain limitations. Primarily, although the framework has been validated on multiple datasets, the inherent complexity and diversity of IoT intrusion detection necessitate further testing on an expanded range of datasets to comprehensively evaluate FSLLM's generalizability. In addition, this study employs fine-tuning of LLMs to generate high-quality samples, which also comes with high computational resource requirements. This means that our data augmentation process can only be performed locally, whereas in many application scenarios, data augmentation often needs to be conducted online. In contrast, some traditional methods (such as SMOTE and undersampling) generate lower-quality data but require significantly fewer computational resources, making them more suitable for online updating scenarios. Moreover, these methods typically do not rely on GPUs or high-performance computing devices. One of the key purposes of using feature selection algorithms to extract high-quality features is to reduce the computational resources required for fine-tuning large language models. However, the training speed of large language models still lags behind traditional methods, necessitating a trade-off between computational cost and data augmentation quality in practical applications. In light of these limitations, we propose the following avenues for future research: (1) Exploration of methodologies to further reduce computational costs associated with LLMs fine-tuning, including the optimization of model architectures and training strategies, as well as the utilization of distributed computing resources. (2) Integration of the FSLLM framework with advanced technologies such as federated learning and edge computing to augment its applicability and reliability in complex IoT environments.

Data availability

The datasets used and analyzed during the current study are publicly available as the following: https://staff.itee.uq.edu.au/marius/NIDS_datasets/.

Received: 10 August 2024; Accepted: 24 June 2025
Published online: 01 July 2025

References

1. Mishra, R. & Mishra, A. Current research on Internet of Things (IoT) security protocols: A survey. *Comput. Security*. 104310 https://doi.org/10.1016/j.cose.2024.104310 (2025).
2. Rabbani, M. et al. A lightweight IoT device identification using enhanced behavioral-based features. *Peer-to-Peer Netw. Appl.* **18**, 1–22. https://doi.org/10.1007/s12083-024-01891-9 (2025).
3. Bala, B. & Behal, S. AI techniques for IoT-based DDoS attack detection: Taxonomies, comprehensive review and research challenges. *Comput. Sci. Rev.* **52**, 100631. https://doi.org/10.1016/j.cosrev.2024.100631 (2024).
4. Sun, C.-C., Hahn, A. & Liu, C.-C. Cyber security of a power grid: State-of-the-art. *Int. J. Electr. Power Energy Syst.* **99**, 45–56. https://doi.org/10.1016/j.ijepes.2017.12.020 (2018).
5. Shahin, M., Maghanaki, M., Hosseinzadeh, A. & Chen, F. F. Advancing network security in industrial IoT: a deep dive into AI-enabled intrusion detection systems. *Adv. Eng. Inform.* **62**, 102685. https://doi.org/10.1016/j.aei.2024.102685 (2024).
6. Meziane, H. & Ouerdi, N. A survey on performance evaluation of artificial intelligence algorithms for improving IoT security systems. *Sci. Rep.* **13**, 21255. https://doi.org/10.1038/s41598-023-46640-9 (2023).
7. Termos, M. et al. GDLC: A new graph deep learning framework based on centrality measures for intrusion detection in IoT networks. *Internet of Things* **26**, 101214. https://doi.org/10.1016/j.iot.2024.101214 (2024).
8. Sarhan, M. et al. Doc-nad: A hybrid deep one-class classifier for 613 network anomaly detection. In *2023 IEEE/ACM 23rd Int. Symp. on Clust. Cloud Internet Comput. Work. (CCGridW)* 1–7, https://doi.org/10.1109/ccgridw59191.2023.00016 (2022).
9. Sarhan, M., Layeghy, S. & Portmann, M. Feature analysis for machine learning-based IoT intrusion detection. *arXiv preprint* arXiv:2108.12732 https://doi.org/10.21203/rs.3.rs-2035633/v1 (2021).
10. Khammassi, C. & Krichen, S. A GA-LR wrapper approach for feature selection in network intrusion detection. *Comput. Security* **70**, 255–277. https://doi.org/10.1016/j.cose.2017.06.005 (2017).

11. Subramani, S. & Selvi, M. Multi-objective PSO based feature selection for intrusion detection in IoT based wireless sensor networks. *Optik* **273**, 170419. https://doi.org/10.1016/j.ijleo.2022.170419 (2023).
12. Sarhan, M., Layeghy, S., Moustafa, N., Gallagher, M. & Portmann, M. Feature extraction for machine learning-based intrusion detection in IoT networks. *Digit. Commun. Netw.* https://doi.org/10.1016/j.dcan.2022.08.012 (2022).
13. Yang, R. et al. Efficient intrusion detection toward IoT networks using cloud-edge collaboration. *Comput. Netw.* **228**, 109724. https://doi.org/10.1016/j.comnet.2023.109724 (2023).
14. Talukder, M. A. et al. Machine learning-based network intrusion detection for big and imbalanced data using oversampling, stacking feature embedding and feature extraction. *J. Big Data* **11**, 33. https://doi.org/10.1186/s40537-024-00886-w (2024).
15. Sayegh, H. R., Dong, W. & Al-madani, A. M. Enhanced intrusion detection with LSTM-based model, feature selection, and smote for imbalanced data. *Appl. Sci.* **14**, 479. https://doi.org/10.3390/app14020479 (2024).
16. Ding, S., Kou, L. & Wu, T. A GAN-based intrusion detection model for 5g enabled future metaverse. *Mob. Netw. Appl.* **27**, 2596–2610. https://doi.org/10.1007/s11036-022-02075-6 (2022).
17. Snoussi, R. & Youssef, H. VAE-based latent representations learning for botnet detection in IoT networks. *J. Netw. Syst. Manag.* **31**, 4. https://doi.org/10.1007/s10922-022-09690-4 (2023).
18. Majhi, B. *et al.* Optimizing LightGBM for intrusion detection systems using GOA. In *2023 14th International Conference on Computing Communication and Networking Technologies (ICCCNT)*, 1–5 https://doi.org/10.1109/icccnt56998.2023.10308360 (2023).
19. Shaker, B. N., Al-Musawi, B. Q. & Hassan, M. F. A comparative study of ids-based deep learning models for IoT network. In *Proceedings of the 2023 International Conference on Advances in Artificial Intelligence and Applications*, 15–21 https://doi.org/10.1145/3603273.3635058 (2023).
20. Manocchio, L. D. et al. Flowtransformer: A transformer framework for flow-based network intrusion detection systems. *Expert Syst. Appl.* **241**, 122564. https://doi.org/10.1016/j.eswa.2023.122564 (2024).
21. Karthikeyan, M., Manimegalai, D. & RajaGopal, K. Firefly algorithm based WSN-IoT security enhancement with machine learning for intrusion detection. *Sci. Rep.* **14**, 231. https://doi.org/10.1038/s41598-023-50554-x (2024).
22. Nguyen, H. & Kashef, R. TSI-IDS: Traffic-aware self-supervised learning for IoT network intrusion detection. *Knowl.-Based Syst.* **279**, 110966. https://doi.org/10.1016/j.knosys.2023.110966 (2023).
23. Wang, Y. et al. N-STGAT: Spatio-temporal graph neural network based network intrusion detection for near-earth remote sensing. *Remote Sens.* **15**, 3611. https://doi.org/10.3390/rs15143611 (2023).
24. Leevy, J. L., Hancock, J. T., Khoshgoftaar, T. M. & Peterson, J. M. IoT information theft prediction using ensemble feature selection. *J. Big Data* https://doi.org/10.1186/s40537-021-00558-z (2022).
25. Mohy-Eddine, M., Guezzaz, A., Benkirane, S., Azrour, M. & Farhaoui, Y. An ensemble learning based intrusion detection model for industrial IoT security. *Big Data Min. Anal.* **6**, 273–287. https://doi.org/10.26599/bdma.2022.9020032 (2023).
26. Komisarek, M., Pawlicki, M., Kozik, R., Hołubowicz, W. & Choraś, M. How to effectively collect and process network data for intrusion detection?. *Entropy* **23**, 1532. https://doi.org/10.3390/e23111532 (2021).
27. Amin, R., El-Taweel, G., Ali, A. F. & Tahoun, M. Hybrid chaotic zebra optimization algorithm and long short-term memory for cyber threats detection. *IEEE Access* https://doi.org/10.1109/access.2024.3397303 (2024).
28. Mouiti, M., Elhariri, A., Habibi, O. & Lazaar, M. Toward improving internet of things (IoT) networks security using machine learning based intrusion detection system. In *2023 International Conference on Digital Age & Technological Advances for Sustainable Development (ICDATA)*, 46–51 https://doi.org/10.1109/icdata58816.2023.00018 (2023).
29. Liu, J., Gao, Y. & Hu, F. A fast network intrusion detection system using adaptive synthetic oversampling and LightGBM. *Comput. Security* **106**, 102289. https://doi.org/10.1016/j.cose.2021.102289 (2021).
30. Soflaei, M. R. A. B., Salehpour, A. & Samadzamini, K. Enhancing network intrusion detection: a dual-ensemble approach with CTGAN-balanced data and weak classifiers. *J. Supercomput.* 1–33, https://doi.org/10.1007/s11227-024-06108-7 (2024).
31. Wang, C. *et al.* Classification of IoT intrusion detection data based on WGAN-gp and E-GraphSAGE. In *Third International Conference on Green Communication, Network, and Internet of Things (CNIoT 2023)*, vol. 12814, 288–292 https://doi.org/10.1117/12.3010362. (2023).
32. Li, F. et al. Pre-trained language model-enhanced conditional generative adversarial networks for intrusion detection. *Peer-to-Peer Netw. Appl.* **17**, 227–245. https://doi.org/10.1007/s12083-023-01595-6 (2024).
33. Peng, H., Long, F. & Ding, C. Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy. *IEEE Trans. Pattern Anal. Mach. Intell.* **27**, 1226–1238. https://doi.org/10.1109/tpami.2005.159 (2005).
34. Hansen, N., Müller, S. D. & Koumoutsakos, P. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). *Evol. Comput.* **11**, 1–18. https://doi.org/10.1162/106365603321828970 (2003).
35. Ramírez-Gallego, S. et al. Fast-mRMR: Fast minimum redundancy maximum relevance algorithm for high-dimensional big data. *Int. J. Intell. Syst.* **32**, 134–152. https://doi.org/10.1002/int.21833 (2017).
36. Ke, G. et al. Lightgbm: A highly efficient gradient boosting decision tree. *Adv. Neural. Inf. Process. Syst.* **30**, 3149–3157. https://doi.org/10.5555/3294996.3295074 (2017).
37. Lin, T.-Y., Goyal, P., Girshick, R., He, K. & Dollár, P. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, 2980–2988 https://doi.org/10.1109/iccv.2017.324 (2017).
38. Lübbering, M., Gebauer, M., Ramamurthy, R., Bauckhage, C. & Sifa, R. Decoupling autoencoders for robust one-vs-rest classification. In *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, 1–10 https://doi.org/10.1109/DSAA53316.2021.9564136 (2021).
39. Chang, Y. et al. A survey on evaluation of large language models. *ACM Trans. Intell. Syst. Technol.* **15**, 1–45. https://doi.org/10.1145/3641289 (2023).
40. Ding, B. *et al.* Data augmentation using llms: Data perspectives, learning paradigms and challenges. *arXiv preprint* arXiv:2403.02990 https://doi.org/10.48550/arXiv.2403.02990 (2024).
41. Sennrich, R., Haddow, B. & Birch, A. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 1715–1725 https://doi.org/10.18653/v1/P16-1162 (2016).
42. Sui, Y., Zhou, M., Zhou, M., Han, S. & Zhang, D. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, 645–654 https://doi.org/10.1145/3616855.3635752 (2024).
43. Sarhan, M., Layeghy, S. & Portmann, M. Towards a standard feature set for network intrusion detection system datasets. *Mob. Netw. Appl.* **27**, 1–14. https://doi.org/10.1007/s11036-021-01843-0 (2022).
44. Sarhan, M., Layeghy, S. & Portmann, M. Evaluating standard feature sets towards increased generalisability and explainability of ml-based network intrusion detection. *Big Data Res.* **30**, 100359. https://doi.org/10.1016/j.bdr.2022.100359 (2022).
45. Radford, A. et al. Language models are unsupervised multitask learners. *OpenAI Blog* **1**, 9 (2019).
46. Borisov, V., Sessler, K., Leemann, T., Pawelczyk, M. & Kasneci, G. Language models are realistic tabular data generators. In *The Eleventh International Conference on Learning Representations*, https://doi.org/10.48550/arXiv.2210.06280 (2023).
47. Van Thieu, N. & Mirjalili, S. Mealpy: An open-source library for latest meta-heuristic algorithms in python. *J. Syst. Architect.* **139**, 102871. https://doi.org/10.1016/j.sysarc.2023.102871 (2023).

Acknowledgements
This work was supported by the Major Program of Songshan Laboratory under Grant ZZK202403002, and the Key Scientific and Technological Project of Henan Province under Grant 232102210069.

Author contributions
Huan Ma provided the research methodology. Huan Ma and Wan Zhang conducted most of the experiments, while Wan Zhang completed the initial draft of the manuscript. Dalong Zhang and Baozhan Chen provided constructive feedback for the manuscript revision. All authors read and approved the final manuscript.

Declarations

Competing interests
The authors declare no competing interests.

Additional information
Correspondence and requests for materials should be addressed to H.M.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.