scientific reports

Check for updates

OPEN

A federated transformer-enhanced double Q-network for collaborative intrusion detection

Tianqi Ma, Yabo Yin✉, Wenzhong Yang✉ & Shuya Wang

With the rapid proliferation of IoT technology, cybersecurity challenges have become increasingly prominent. Traditional centralized network intrusion detection systems (NIDS) exhibit significant limitations including privacy risks and inadequate modeling of spatiotemporal correlations. This paper proposes FedT-DQN (Transformer-based Federated Double Q-Network), a novel federated reinforcement learning framework for dynamic intrusion detection. The method incorporates: (1) A Transformer encoder as federated aggregator using self-attention mechanisms; (2) A dual-layer Q-network architecture decomposing detection into feature extraction and decision optimization; (3) Soft Actor-Critic (SAC) integration for local training considering system heterogeneity. Experimental results on four benchmark datasets show that FedT-DQN achieves over 99% detection accuracy with enhanced F1 scores and reduced false positive rates, all while maintaining data privacy. The source code for this study is available at https://github.com/BuLaTaa/FedT-DQN-in-IDS..

Keywords Federated learning, Intrusion detection, Transformer, Double Q-network, Reinforcement learning

With the rapid development of Internet of Things (IoT) technology, its applications have penetrated into key areas such as smart homes, smart cities, industrial automation, healthcare, and intelligent transportation, driving society towards a highly interconnected digital future. This growth has significantly expanded the attack surface for network security, with new threats such as botnets, distributed denial-of-service (DDoS), and zero-day exploits continuing to emerge[1], making cybersecurity a core challenge in achieving trustworthy IoT systems[2,3]. Against this backdrop, machine learning-based network intrusion detection systems (ML-based NIDS) have garnered significant attention due to their ability to uncover potential threats from massive heterogeneous traffic data[4,5]. Traditional network intrusion detection systems (NIDS) focus on signature-based and anomaly-based methods. Signature-based approaches achieve high-precision detection by matching known attack signatures but fail to identify new threats. Anomaly-based methods use machine learning algorithms (such as support vector machines[6,7], random forests[8], convolutional neural networks[9]) to model normal behavior patterns and detect anomalies. However, these approaches face critical limitations:

1. Traditional ML methods heavily rely on manual feature engineering and single-sample analysis, making it difficult to capture the spatio-temporal correlations between network traffic[10,11].
2. Based on CNN/RNN DL models, although they can automatically extract flow-level features, they fail to model the context of complex attack behaviors (such as distributed denial-of-service attacks) due to ignoring network topology and endpoint interaction relationships[12].
3. In addition, the dynamic evolution of attack patterns and the frequent emergence of new threats force models to be continuously retrained to maintain effectiveness. The centralized training paradigm faces serious privacy risks–network traffic often contains sensitive operational information, and direct sharing may lead to the leakage of trade secrets or malicious abuse[13].

Although there has been significant academic progress in deep models, practical deployment still heavily depends on rule-based systems (e.g., Snort). This reliance highlights issues such as poor generalization and difficulty handling concept drift[14,15]. Therefore, there is a need for new frameworks that balance privacy protection, dynamic adaptability, and cross-domain generalization. Federated Learning (FL) allows for collaborative training through local parameter aggregation while maintaining privacy. It has been successfully applied in areas like smart homes[16] and autonomous driving[17]. However, in network intrusion detection, FL faces two main

School of Computer Science and Technology (School of Cyberspace Security), Xinjiang University, Urumqi 830046, China. ✉email: yinyabo@xju.edu.cn; yangwenzhong@xju.edu.cn

challenges: dynamic network environments that require real-time adaptation and node heterogeneity, which causes instability during convergence. To address these, Federated Reinforcement Learning (FRL) combines RL with FL to improve generalization against unknown threats. However, existing FRL methods rely on simple averaging (FedAvg[18]), which overlooks node environmental differences and fails to capture the temporal nature of attacks. This limits both accuracy and robustness.

This paper proposes FedT-DQN (Federated Transformer-based Double Q-Network) addressing three key shortcomings: (1) Traditional FL aggregation cannot capture spatiotemporal correlations in traffic data; (2) Averaging methods cause policy mismatches in heterogeneous environments[19]; (3) Static feature engineering struggles with rapidly evolving attack types.

The contributions of our proposed FedT-DQN model are as follows:

1. We introduce a Transformer encoder as the federated aggregator, dynamically learning contextual dependencies between nodes through the self-attention mechanism, thereby replacing the traditional weighted averaging operation. This method draws inspiration from FedFormer's contextual aggregation idea but further optimizes the calculation of attention weights. This allows it to identify clusters of nodes with similar threat patterns without exposing node metadata (such as network topology, traffic volume), thereby enhancing the relevance of model updates.
2. We design a Double Q-Network structure, decomposing the intrusion detection strategy into two stages:low-level feature extraction and high-level decision optimization. The low-level network learns spatio-temporal features based on local traffic sequences, utilizing the Transformer's temporal modeling capability to capture long-range dependencies. The high-level network then optimizes global defense strategies through federated collaboration, combining the ϵ-greedy exploration mechanism to balance the detection of known attacks and the discovery of unknown threats.
3. We incorporate Soft Actor-Critic (SAC) into the policy network for training on local clients, taking into account factors such as computational capabilities, data distribution diversity, and network conditions to mitigate the impact of system heterogeneity.

To the best of our knowledge, we are the first to introduce Transformer into Federated Reinforcement Learning as an aggregator for the task of intrusion detection. Experiments demonstrate that this framework achieves significant improvements in key metrics such as F1-score and AUC on four commonly used intrusion detection datasets, compared to intrusion detection methods that employ FedAvg[18] aggregation and Independent Q-Learning[20], while maintaining local data privacy. We also conducted comparative tests on the newly released UNSW-MG24 and CIC IIoT 2025 datasets, where our method continued to show excellent performance. Furthermore, it exhibits significant tolerance to training latency for resource-constrained nodes.

Related works

This section reviews the evolution of artificial intelligence methods in Network Intrusion Detection Systems (NIDS) for the Internet of Things, covering the application challenges and innovations of machine learning (ML) and deep learning (DL) technologies. Building on this foundation, it further summarizes the research progress of Federated Reinforcement Learning (FRL), analyzing its potential in privacy protection and dynamic threat detection.

Network intrusion detection

Traditional IDS research employs various ML techniques including k-Nearest Neighbors (KNN)[21], Decision Trees[22], Support Vector Machines (SVM)[6,7], clustering[23,24], Random Forests[8], and Artificial Neural Networks (ANN)[25]. However, these models increase system complexity and reduce efficiency with large heterogeneous data volumes[26].Deep learning-based IDS has evolved through multiple technical approaches. Autoencoders (AE) represent unsupervised learning, with Muhammad et al.[27] applied stacked AEs with multi-layer perceptrons for financial IoT security, validating feature latent space optimization on KDDCup99. Siddiqui[28] proposed adaptive ensemble AE models supporting lightweight edge deployment while maintaining stable performance. Convolutional Neural Networks (CNN) leverage spatial feature extraction advantages. Karaca et al.[29] utilized 1D-CNN with adaptive particle swarm optimization for botnet detection, achieving 98% accuracy but risking overfitting. For temporal dependencies, Le et al.[30] employed RNN variants (LSTM/GRU) for time-based traffic features, reducing detection error by 2% on NSL-KDD. Ullah's team[31] proposed lightweight LSTM-GRU models using regularization and weighted loss for data imbalance. Recent hybrid models gain popularity by fusing multi-perspective features. Wang et al.[32,33]combined Transformer with CNN, achieving 1.87% accuracy improvement on CICDDoS2019, though generalization was limited by single dataset validation. Hassan[34] innovatively fused CNN with Weight-Dropped LSTM, using CNN for feature extraction and WDLSTM for dependency modeling, effectively alleviating overfitting on UNSW-NB15. Table 1 compares performance and limitations of various IDS models using different ML algorithms. However, existing methods face challenges including high complexity, low training efficiency, and insufficient privacy protection, requiring exploration of lightweight architectures, dynamic attention mechanisms, and federated learning integration.As an expansion to IDS applications, M. Koca[35,36] provides supplementary insights into machine learning-based methods for network security risk detection and adaptive modeling.

Federated learning-based IDS

Table 2 summarizes relevant work utilizing Federated Learning (FL) for Network Intrusion Detection (NIDS). However, existing work has several limitations. Most FL research relies solely on FedAvg aggregation[43], which struggles with real-world data heterogeneity and robustness issues. Additionally, commonly used CNNs and

Refs.	Algo.	Dataset	Acc. (%)	Limitation
[37]	SVM	UNSW-NB15	84.66	It's hard to handle large-scale network traffic data.
[38]	DT	NSL-KDD	90.30	Poor generalization ability.
[39]	ANN	Self gen.	97.51	Difficulty in gradient propagation.
[40]	NAV	Kyoto	97.60	Network features often exhibit strong correlations with each other.
[41]	CNN	CICIDS2017	98.56	Network features often lack the spatial locality characteristics of image data.
[42]	RNN	CICIDS2017	95.24	Computational efficiency is low.

Table 1. A comparison of existing research on intrusion detection using machine learning and deep learning.

Refs.	ML Algorithm	Dataset	Aggregation Function	non-IID	Generalization
Stacked-Unsupervised[43]	Stacking Deep Autoencoder	TON_IoT, Bot-IoT, CICIDS2018, UNSW-NB15	FedAvg		✓
VAN-IDS[47]	LightGBM	Vein	FedAvg	✓	
FELIDS[44]	DNN, CNN, RNN	MQTTset	Fed+		
FL for Industrial IoT[45]	DNN	CIC-ToN_IoT	FedAvg, Fed+	✓	
FDRL-IDS[46]	Q-net	NSL-KDD, ISOT-CID	Majority Vote		
Ours	Q-Network, Transformer	TON_IoT-v2, Bot-IoT-v2, CICIDS2018-v2 UNSW-NB15-v2 UNSW-MG24 CIC IIoT 2025	Dynamic Contextual weights	✓	✓

Table 2. Intrusion detection systems based on federated learning.

DNNs[44,45] face significant challenges: CNNs are ill-suited for tabular network traffic data, while DNNs suffer from overfitting and class imbalance problems, often biasing toward majority classes and failing to detect subtle attack behaviors.Few studies simultaneously address data privacy protection, model scalability, and generalization. While some incorporate reinforcement learning methods like Deep Q-Learning (DQN)[46], these approaches struggle to capture long-distance temporal dependencies in local feature extraction.Therefore, this study employs Transformer self-attention mechanisms and Q-network reinforcement learning within federated environments. Transformers efficiently model global dependencies between network flow features, while Q-networks optimize detection strategies through reward mechanisms. We evaluate various non-IID scenarios and implement dynamic attention-weighted averaging aggregation beyond traditional FedAvg.

Background

In this section, we will discuss the relevant background knowledge of federated learning, reinforcement learning, and attention mechanisms.

Federated learning

Traditional machine learning algorithms rely on centralized data storage, requiring all training data to be collected in a single data center. However, with increasing demands for data privacy protection, this centralized model faces significant challenges, particularly in sensitive fields such as healthcare and finance where data leakage risks are critical. Federated Learning (FL) emerges as an innovative distributed collaborative learning paradigm that constructs a decentralized training framework by integrating machine learning, distributed computing, and privacy protection technologies. Its core mechanism allows multiple data holders (clients) to jointly optimize a global model without exchanging original local data through parameter transmission. Each participating client trains the model using local data, periodically uploads model parameters to the central server for global aggregation, then synchronizes the optimized global parameters to update local models. This "data stays local, model moves" paradigm effectively reduces bandwidth demands while fundamentally avoiding data privacy leakage through local data storage. The communication process follows these steps:

1. Global model initialization: The system selects a subset of participating gateways according to a preset ratio, initializes the main model, and distributes it to all selected gateways to initiate distributed training.
2. Local model training: Each gateway performs localized training on the received global model using its private dataset, generating an enhanced sub-model adapted to local data characteristics through iterative optimization.
3. Parameter feedback: All participating devices upload optimized model weights to the central server through encrypted channels to ensure data privacy.

4. Federated model aggregation: The central server employs a weighted averaging aggregation algorithm to fuse distributed model parameters, integrating learning outcomes from each node to construct a comprehensive global model.
5. Model synchronization: The enhanced global model is distributed to all client devices via secure transmission protocol, completing the federated learning loop and improving generalization capability across different data domains.

Reinforcement learning

Reinforcement Learning (Reinforcement Learning, RL), as an important paradigm in the field of machine learning, is committed to training intelligent agents (Agent) to autonomously construct optimal decision-making strategies through interactive learning with dynamic environments (Environment). Its core mechanism is embodied in the "action-feedback-learning" closed loop: after the agent performs an action (Action), it receives immediate rewards (Reward) from the environment's feedback, continuously improving its decision-making ability through policy iteration optimization. At the level of mathematical modeling, reinforcement learning problems are usually formalized as Markov Decision Processes (Markov Decision Process, MDP)[48], whose five-element framework includes:

1. State Space: A complete representation of the observable features of the environment.
2. Action Space: The set of controllable operations for the agent.
3. Reward Function: The environment's immediate evaluation of state-action pairs.
4. Transition Dynamics: The stochastic rules governing the evolution of the environment's state.
5. Discount Factor: The decay coefficient for future rewards.

The Policy Function, as the core decision-making component of the agent, defines the probability distribution mapping from states to actions. Its implementation ranges from table lookup methods based on dynamic programming to more complex frameworks such as Deep Q-Networks (DQN). The Value Function, on the other hand, estimates the long-term expected return of states using the Bellman Equation, providing a theoretical basis for policy optimization. The learning objective can be formally expressed as:

$$V^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^{t} R(s_t, a_t) | s_0 = s\right] \tag{1}$$

Where $\gamma \in [0, 1)$ is the temporal discount factor, and $V^{\pi}(s)$represents the long-term value of state ss under policy π. Through methods such as Temporal Difference Learning (TDLearning) or Policy Gradient, the agent continuously updates its policy parameters, ultimately converging to the optimal policy π^{*}.

Deep Q-learning

Q-learning represents a value-based reinforcement learning approach that iteratively refines state-action estimates to determine optimal policies. The algorithm progressively enhances Q-values using Bellman optimality principles through temporal difference learning:

$$Q^{new}(s_t, a_t) \quad = Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)\right] \tag{2}$$

Parameters α (learning coefficient) and γ (discount factor) control update magnitude and future reward importance. Traditional Q-learning faces scalability limitations in discrete, finite state-action spaces. Deep Q-Learning addresses these constraints by employing neural networks as function approximators, enabling Q-value prediction for high-dimensional state representations. The optimization minimizes mean squared error between predicted and target values:

$$\text{loss} = \left(r + \gamma \max_{a'} \hat{Q}(s, a') - Q(s, a)\right)^2 \tag{3}$$

Experience replay enhances stability by breaking data correlations, while Prioritized Experience Replay improves sampling efficiency through dynamic priority weighting based on temporal difference errors. This approach enables effective handling of complex state spaces for practical applications in network intrusion detection.

Soft Actor-Critic (SAC)

Policy gradient approaches in reinforcement learning optimize parameters θ within policy π_θ through value function-guided parameter updates:

$$\nabla_\theta J^{\pi_\theta}(s_t) = \nabla_\theta log(\pi_\theta(a_t|s_t)) \sum_{t'=t}^{\infty} \gamma^{t'-t} R(s_{t'}, a_{t'}) \tag{4}$$

This methodology encounters significant variance challenges stemming from cumulative reward summations across episodes, where reward distributions exhibit substantial fluctuations between different trajectories. Actor-critic architectures address this limitation by employing value approximation functions that estimate expected returns while undergoing independent optimization processes. The critic component, representing the Q-function, utilizes off-policy temporal difference learning for error minimization:

$$y = r(s,a) + \gamma \mathbb{E}_{a' \approx \pi(s')} Q_{\hat{\psi}(s',a')} \tag{5}$$

The target Q-function Q_{ψ}represents a parameterized average of historical Q-functions, while D denotes the experience replay mechanism for past interactions. Contemporary actor-critic variants incorporate[49] entropy regularization terms within policy gradient computations to enhance exploration capabilities. The Soft Actor-Critic (SAC) framework implements this enhancement through modified policy updates:

$$\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim D, a\sim\pi}\Big[\nabla_\theta \log \pi_\theta(a|s) \times \big(- a \log \pi_\theta(a|s) + Q_\psi(s,a) - b(s)\big)\Big] \tag{6}$$

Where b(S) is a learned state-dependent baseline function.

Self-attention mechanism
Self-Attention is a mechanism that models global dependencies by dynamically calculating the correlation weights between elements within a sequence. Its core formula is:

$$\text{Attention}(Q,K,V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{7}$$

Here the input sequence X=$[x_1, x_2, \ldots, x_n]$ generates the query matrix $Q = XW_Q$,$Q = XW_Q$, the key matrix $K = XW_K$, and the value matrix $V = XW_V$ (where W_Q,W_K,W_V are learnable parameter matrices) through linear transformations. After calculating similarity via dot product, a scaling factor of $\sqrt{d_k}$ (where $\sqrt{d_k}$ is the dimension of the key vector) is introduced to stabilize gradients. Finally, the value vectors are weighted and summed to produce context-aware outputs. This mechanism breaks through the local dependency limitations of traditional RNNs, enabling the model to capture associations among all positions within the sequence simultaneously.

Methodology

This section presents a comprehensive methodology framework for assessing intrusion detection system generalization capabilities within diverse network environments. We define heterogeneous networks through datasets derived from distinct network infrastructures, enabling systematic evaluation of cross-domain performance. Existing Transformer-based NIDS models mainly use self-attention mechanisms for feature extraction but rely on traditional parameter aggregation methods (e.g., FedAvg[18]), which cannot effectively address data heterogeneity and dynamic attack patterns. In contrast, the FedT-DQN model introduces Transformer into the federated learning framework and combines it with the Double Q-Network (DQN) architecture from deep reinforcement learning. This model captures long-term dependencies during feature extraction and optimizes the global defense strategy through reinforcement learning. Additionally, FedT-DQN innovatively uses a dynamic self-attention aggregation algorithm to weight detection experiences from different clients, improving adaptability and accuracy in heterogeneous network environments.

The architectural design draws inspiration from established reinforcement learning applications in Meta-World fall detection scenarios and vehicle control balance systems[50]. These precedent studies employed reinforcement learning frameworks for continuous prediction tasks, which we adapted by implementing discrete task prediction networks specifically tailored for intrusion detection scenarios. This methodological adaptation investigates whether discrete prediction mechanisms can strengthen detection capabilities against previously unseen attack vectors. Research evidence supports the generalization potential of this approach[46]. While federated learning presents clear advantages over traditional supervised learning paradigms for NIDS applications (as outlined in Section 1), current research exhibits limited investigation into heterogeneous data scenarios. This research gap results in substantial performance variations when systems encounter diverse dataset characteristics. Furthermore, existing methodologies demonstrate insufficient contextual awareness when confronting novel threat landscapes. As established in Section 2, the majority of existing research lacks comprehensive evaluation under cross-dataset validation frameworks.

Data description

Our experimental framework necessitates uniform feature representation across all federated learning clients to ensure compatibility within horizontal federation architectures. This constraint guided our selection of four benchmark datasets that utilize standardized NetFlow feature extraction protocols. The experimental evaluation employs four IoT security datasets: NF-BoT-IoT-v2, NF-ToN-IoT-v2, NF-CSE-CIC-IDS2018-v2, and NF-UNSW-NB15-v2. Each dataset represents a NetFlow-processed version derived from their corresponding base collections (BoT-IoT, ToN-IoT, CSE-CIC-IDS2018, and UNSW-NB15). All datasets maintain consistent feature dimensionality through 43 flow-level NetFlow attributes. These attributes capture network communication patterns using the standard five-tuple specification: source/destination IP addresses, source/destination ports, and protocol identifiers. The feature space consists entirely of numerical values, enabling direct compatibility across different network environments. While our implementation utilizes NetFlow-standardized versions for consistency, comprehensive dataset characteristics are detailed in Table 3. In the experimental section, we compare the performance of FedT-DQN with other models on two real-world datasets: the UNSW-MG24 microgrid dataset and the CIC IIoT 2025 dataset, which consists of sensor and network data synchronously collected in an Industrial Internet of Things (IIoT) environment.

Dataset	Label type and attack classes	Samples
NF-BoT-IoT	Binary: Normal	13,859
	Binary: Attack	586,241
	Multi: Benign	13,859
	Multi: Theft	1909
	Multi: DoS	56,833
	Multi: DDoS	56,844
	Multi: Reconnaissance	470,655
NF-ToN-IoT	Binary: Normal	270,279
	Binary: Attack	1,108,995
	Multi: Benign	270,279
	Multi: Ransomware	142
	Multi: MITM	1295
	Multi: DoS	17,717
	Multi: Backdoor	17,247
	Multi: Scanning	21,467
	Multi: XSS	99,944
	Multi: Password	156,299
	Multi: DDoS	326,345
	Multi: Injection	468,539
NF-UNSW-NB15	Binary: Normal	2,295,222
	Binary: Attack	95,053
	Multi: Benign	2,295,222
	Multi: Exploits	31,551
	Multi: Fuzzers	22,310
	Multi: Generic	16,560
	Multi: Reconnaissance	12,779
	Multi: DoS	5794
	Multi: Analysis	2299
	Multi: Backdoor	2169
	Multi: Shellcode	1427
	Multi: Worms	164
NF-CICIDS2018	Binary: Normal	16,635,567
	Binary: Attack	2,258,141
	Multi: Benign	16,635,567
	Multi: DDOS-HOIC	1,080,858
	Multi: DoS-Hulk	432,648
	Multi: DoS-LOIC-HTTP	307,300
	Multi: Bot	143,097
	Multi: Infiltration	116,361
	Multi: SSH-Brute	94,979
	Multi: DoS-GoldenEye	27,723
	Multi: FTP-Brute	25,933
	Multi: DoS-SlowHTTP	14,116
	Multi: DoS-Slowloris	9512
	Multi: Brute-Web	2143
	Multi: DDOS-LOIC-UDP	2112
	Multi: Brute-XSS	927
	Multi: SQL injection	432

Table 3. Statistical analysis of the dataset used in our experiment.

Data preprocessing

A hierarchical preprocessing architecture is employed to process heterogeneous network security datasets (CICIDS2018, Bot-IoT, ToN-IoT, NB15). During the client-level processing phase, each participating party performs independently:

1. Dynamically encode categorical features such as protocol type (PROTOCOL/L7_PROTO): $X_{\text{encoded}} = \phi(v) \in \mathbb{Z}^+$, where ϕ is a mapping function based on value frequencies;
2. Apply Z-score normalization after performing boundary clipping on numerical features:$X_{\text{std}} = \frac{X-\mu}{\sigma}$;
3. Verify the validity of the binary labels $y \in 0, 1$;
4. Split the data in a training set: test set = 4:1 ratio (random seed fixed to 42).

During the federated-level alignment phase, the system aggregates the global feature space $\mathscr{F}g = \bigcup k = 1^K \mathscr{F}k(K$ is the number of clients), performs zero-padding for missing features, and ensures that all client feature matrices satisfy $\mathrm{X}_k \in \mathbb{R}^{n \times d_g}$ ($d_g = |\mathscr{F}_g|$) through column sorting. This process successfully eliminates the heterogeneity problem between datasets where $d_k \neq d_j$, providing the model with dimensionally unified inputs $\mathrm{X}_k, \mathrm{y}kk = 1^K$ while maintaining the federated principle of localizing $\mathscr{D}_k$.

Agent algorithm

Our approach adopts a federated learning architecture where each agent initializes a Q-network consisting of outer encoder, local encoder, transformer encoder, and output decoder networks. We modify the Soft Actor-Critic (SAC) algorithm using double Q-networks for stability and incorporate a contextual federated Q-function. The Local Encoder extracts low-level features, while the Transformer Encoder performs decision optimization. This integrates the experience replay mechanism from Mnih et al.[51], achieving temporal correlation elimination through Monte Carlo sampling on actions based on policy network distribution ($\pi(a|s)$). Experience quadruples (s_t, a_t, r_{t+1}, s_{t+1}) are stored in the experience pool, with random mini-batch sampling[52] breaking Markov chain correlation. The intrusion detection problem is modeled as an MDP where each client maintains an IntrusionDetectionEnv with state space defined by network traffic features and binary action space 0: Normal, 1: Anomalous. The ClassificationPolicy uses a three-layer fully connected network (hidden dimensions: 256) outputting probability distribution through Categorical distribution. The reward function assigns 2.0 for correct attack detection, 1.0 for correct normal identification, and -3.0 penalty for false negatives. Double Q-networks (qf1, qf2) prevent value function overestimation, with input concatenating observed features and one-hot encoded actions. SAC balances exploration and exploitation through entropy regularization $-\alpha \log(\pi_\theta(a|s))$. Policy updates follow: $\nabla_\theta J(\pi_\theta) = \mathbb{E}_{s\sim\mathscr{D},a\sim\pi}\left[\nabla_\theta \log(\pi_\theta(a|s)) \quad \cdot(-\alpha \log(\pi_\theta(a|s)) + Q_\psi(s,a) - b(s))\right]$Q-function updates use temporal difference learning: $\mathscr{L}_Q(\psi) = \mathbb{E}_{(s,a,r,s')\sim\mathscr{D}}\left[(Q_\psi(s,a) \quad -(r+\gamma\mathbb{E}_{a'\sim\pi}[Q_{\bar{\psi}}(s',a')]))^2\right]$

with target network $Q_{\bar{\psi}}$ for stability.To address class imbalance, SimpleReplayBuffer implements weighted cross-entropy loss with normal samples weighted 1.0 and attack samples 10.0: $\mathscr{L}cls = \mathrm{CrossEntropyLoss}(\mathrm{logits}, \mathrm{labels}, \mathrm{weight} = [1.0, 10.0])$. Policy loss combines classification and Q-value maximization: $\mathscr{L}policy = -\mathbb{E}[Q(s,a)] + 0.5 \cdot \mathscr{L}cls$. Training uses FedPathCollector for data collection, storing ($observation$, $action$, $reward$, $next_o bservation, terminal, labels$) tuples. Parameter synchronization employs soft updates $\theta_{target} \leftarrow \tau\theta + (1-\tau)\theta_{target}$ ($\tau = 0.005$) for convergence stability. Shown as Algorithm 1.

Require: Traffic features $\mathscr{X}$, labels $\mathscr{Y}$
Ensure: Updated policy π_θ, Q-networks Q_{ψ_1}, Q_{ψ_2}
Initialize: Networks π_θ, Q_{ψ_1}, Q_{ψ_2}, targets $Q_{\bar{\psi}_j}$, buffer $\mathscr{D}$
for episode $t = 1, \ldots, T$ **do**
  for step $k = 0, \ldots, K-1$ **do**
    Sample action $a_k \sim \pi_\theta(\cdot|s_k)$
    Compute reward r_k based on detection accuracy
    Store $(s_k, a_k, r_k, s_{k+1}, d_k, y_k)$ in $\mathscr{D}$
  end for
  if $|\mathscr{D}| \geq$ batch_size **then**
    Sample batch $\{(s_i, a_i, r_i, s'_i, d_i, y_i)\}_{i=1}^B \sim \mathscr{D}$
    Update Q-networks: $\mathscr{L}_{Q_j} = \mathbb{E}[(Q_{\psi_j}(s,a) - y)^2]$
    Update policy: $\mathscr{L}_\pi = \mathbb{E}[\alpha \log \pi_\theta - Q_\psi] + \lambda \mathscr{L}_{\text{cls}}$
    Soft update targets: $\bar{\psi}_j \leftarrow \tau\psi_j + (1-\tau)\bar{\psi}_j$
  end if
end for

Algorithm 1. Client-side SAC for Intrusion Detection

Require: Local encoders $\{E_i\}_{i=1}^{N}$, observation-action (o, a)
Ensure: Federated Q-value $Q_{\text{fed}}(o, a)$
Initialize: Transformer $\mathscr{T}$, decoder D
for round $r = 1, \ldots, R$ **do**
  Generate embeddings: $e_i = E_i(\text{concat}(o, \text{OneHot}(a))) + PE_i$
  Construct sequence: $\mathscr{E} = [\text{CLS}, e_1, \ldots, e_N]$
  Apply multi-head attention: $\mathscr{E}' = \text{Transformer}(\mathscr{E})$
  Extract global context: $h_{\text{global}} = \mathscr{E}'[0]$
  Compute Q-value: $Q_{\text{fed}}(o, a) = D(h_{\text{global}} \oplus E_i(o, a))$
  Exchange parameters and check convergence
end for

Algorithm 2. Federated Aggregation with Transformer

Global model aggregation

The core innovation of this study lies in the FedT-DQN federated knowledge aggregation strategy based on Transformer attention mechanism, overcoming limitations of traditional FedAvg parameter averaging. Each client maintains N encoder networks: one trainable local encoder and N-1 frozen external encoders from other clients. During federated aggregation, the system inputs observation-action pair (o,a) into all encoders generating feature embeddings: $e_i = E_i(o \oplus \text{onehot}(a))$, where action a is one-hot encoded and concatenated with observation features. To enable Transformer source distinction, learnable client identity encoding is added: $e_i' = e_i + \text{PositionalEncoding}(i)$. A special CLS token serves as global context representation carrier. The Transformer encoder calculates inter-client correlation weights using multi-head attention: $\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$, where query, key, and value matrices are $Q = EW_Q$, $K = EW_K$, $V = EW_V$, and E is the input sequence containing all client embeddings and CLS token. Multi-head attention captures complementary information between clients from different representation subspaces, with each head focusing on specific knowledge patterns. The two-layer Transformer architecture learns basic inter-client correlations in the first layer and performs higher-order knowledge fusion in the second layer. Dynamic attention weights enable adaptive weighting of different clients' detection experiences based on current network state, demonstrating strong adaptability for emerging attacks and heterogeneous environments. Shown as Algorithm 2.

Federated Q-function calculation integrates global context with local features: $Q_i(o, a) = \text{Decoder}(\text{concat}(\text{CLS}, \text{Encoder}_{\text{local}}(o, a)))$. The CLS token contains aggregated knowledge from all clients after Transformer processing, while local encoder output preserves client-specific domain features. The decoder employs three-layer fully connected architecture mapping concatenated features of dimension 2×256 to scalar Q-value output. This design leverages collective federated intelligence while maintaining local specialization. During backpropagation, only local encoder and Transformer aggregation network parameters update, keeping external encoders frozen to prevent gradient conflicts and ensure stability. The federated aggregation protocol through FedAlgorithm class facilitates periodic knowledge exchange and model synchronization. Clients upload only local encoder parameters rather than entire models, protecting data privacy with lower communication overhead than traditional FedAvg. The server distributes updated encoder parameters to all clients, updating external encoder sets. The system implements early stopping based on F1 score and best model saving, automatically terminating training when global F1 score fails to improve for 5 consecutive rounds, preventing overfitting. This mechanism achieves intelligent knowledge fusion while maintaining personalized client capabilities, providing an ideal solution for collaborative intrusion detection in heterogeneous networks. Algorithm 2 outlines the aggregation process.

The integrated FedT-DQN model

The FedT-DQN framework integrates Federated Learning (FL) with Transformer-based aggregation and Deep Q-Networks (DQN) to enhance the performance and privacy of intrusion detection systems (IDS) in dynamic IoT environments. The key advantage of FedT-DQN lies in its ability to handle heterogeneous data from multiple clients and its robustness against non-IID data and adversarial attacks. The system architecture consists of multiple clients, each responsible for local training, and a central server aggregating model updates from clients using a self-attention-based Transformer mechanism.

Each client trains its local model using Q-learning to predict optimal actions in real-time, while a Double Q-Network (DQN) is employed to mitigate Q-value overestimation, a common issue in standard Q-learning. Furthermore, the model utilizes Soft Actor-Critic (SAC) to optimize the defense strategy dynamically, ensuring robust adaptation to evolving attack patterns. The global model is updated using federated aggregation methods that ensure both privacy and efficiency by only sharing model weights (or Logits) instead of raw data.

The Algorithm 3 describes the key steps in the FedT-DQN framework:

Require: Traffic features $\mathscr{X}$, labels $\mathscr{Y}$, Federated model update parameters
Ensure: Updated global model with aggregated knowledge from clients
Initialize global model π_0, Q-networks Q_1, Q_2, SAC targets Q_θ, buffer $\mathscr{D}$
for each episode $i = 1, \ldots, T$ **do**
  for each client $k = 1, \ldots, K$ **do**
    Train local model using client-specific data $\mathscr{X}_k$ and labels $\mathscr{Y}_k$
    Compute action $a_k \sim \pi_0(s_k)$ using self-attention-based aggregation (Transformer)
    Compute reward r_k based on detection accuracy (e.g., F1-score, AUC)
    Update local Q-values with DQN and policy with SAC optimization
    Store transition in buffer: $(s_k, a_k, r_k, s_{k+1}, y_k)$
  end for
end for
if $|\mathscr{D}| \geq$ batch_size **then**
  Sample batch $\{(s_i, a_i, r_i, s_{i+1}, d_i)\}_{i=1}^{B} \sim \mathscr{D}$
  Update Q-networks: $L_Q = \mathbb{E}\left[(Q_\theta(s,a) - y)^2\right]$
  Update policy: $L_\pi = \mathbb{E}[\alpha \log \pi_\theta(a|s) - Q_\theta(s,a)]$
  Soft update targets: $\theta' = \tau\theta + (1-\tau)\theta'$
end if

Algorithm 3. FedT-DQN Framework for Intrusion Detection

with the framework presented in Fig. 1.

Experimental design

We constructed an evaluation framework incorporating four benchmark datasets–NF-UNSW-NB15-v2, NF-CSE-CIC-IDS-2018-v2, NF-Bot-IoT-v2, and NF-ToN-IoT-v2–to validate the detection efficacy of the proposed architecture. The experiments employed a dual-validation strategy for performance assessment. In the heterogeneous data distribution experiment, each client was configured with a distinct dedicated dataset,

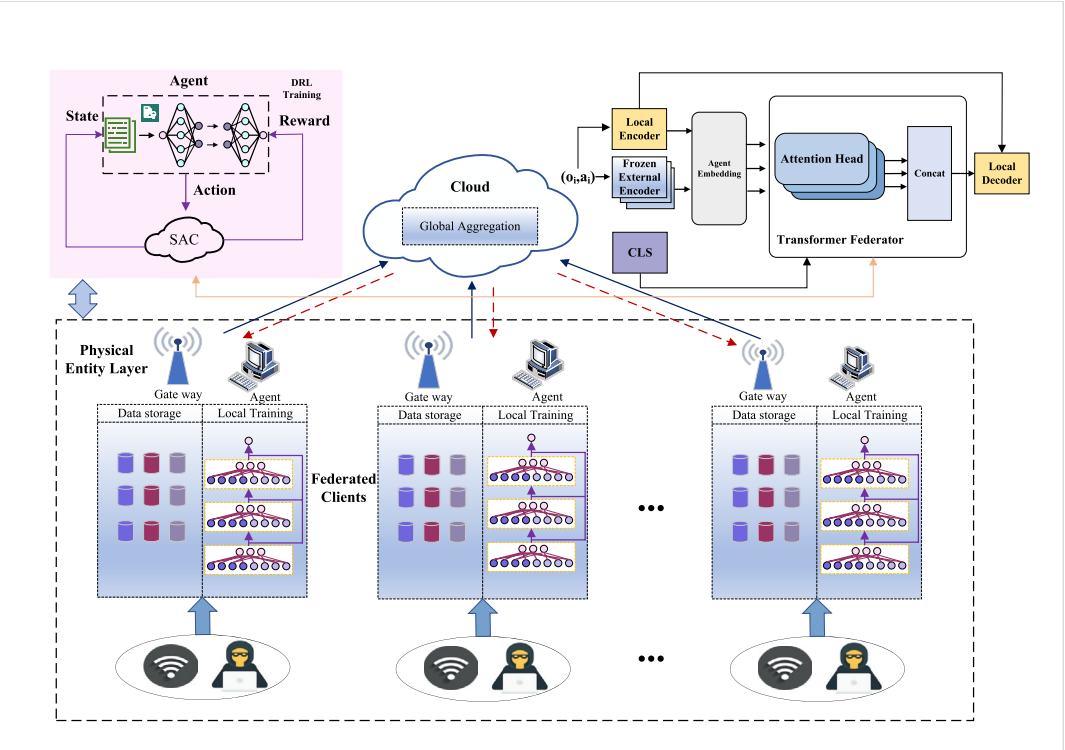

Fig. 1. The framework of FedT-DQN.SAC provides the local reinforcement learning training framework, employing the Double Q-network for stability, while the Transformer is integrated as the core of federated aggregation within the Q-network itself.

simulating real-world data heterogeneity and privacy constraints. In the homogeneous data distribution experiment, the complete dataset was randomly partitioned equally among all participating nodes to establish a performance baseline for data-sharing scenarios. We deployed a federated intrusion detection architecture comprising four edge agent nodes and a central aggregation server. System performance was quantitatively evaluated using standard classification metrics: accuracy, precision, recall, F1-score, and AUC.

This study employs a federated learning architecture with four clients, each utilizing a distinct intrusion detection dataset (NF-CIC-IDS2018-v2, NF-Bot-IoT-v2, NF-ToN-IoT-v2, NF-UNSW-NB15-v2). This setup effectively simulates the data heterogeneity found in real-world network environments. The selection of the number of clients is based on the following considerations: First, four clients provide sufficient data diversity to validate the effectiveness of federated learning while maintaining experimental control and reproducibility. Second, this scale aligns with the typical configurations of small and medium-sized network alliances in real-world deployment scenarios. The number of training rounds is set to 200, combined with an early stopping mechanism (patience=5), which ensures the model's sufficient convergence while preventing overfitting. Each training round includes 1,000 steps of exploration sampling and 100 gradient updates. This setup balances the computational overhead between data collection and model training, ensuring that each client has sufficient interaction data to update its strategy.Other detailed parameters are shown in Table 4.

To implement our work, our experiments were parallelly run on a computing cluster consisting of an Intel® Xeon® Gold 5318Y CPU and 8 A40 GPUs, depending on availability.

Standard machine learning metrics, such as Accuracy, Precision, F1-Score, and Recall, are used to evaluate the model performance of our proposed system. These metrics are derived from the following values:True Positive, True Negative, False Positive and False Negative.

1. TP:True Positive is the number of outcomes in which the model correctly predicts the positive class.
2. TN:True Negative is the number of outcomes in which the mode correctly predicts the negative class.
3. FP:False Positive is the number of outcomes in which the model incorrectly predicts the positive class.
4. FN:False Negative is the number of outcomes in which the model incorrectly predicts the negative class.

The different metrics can be calculated from the items above as follows:

- **Accuracy**: It is the ratio of the total number of correct predictions to the number of total predictions. $Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$
- **Precision**: It is the ratio of the number of correctly predicted positive samples to the number of samples that are predicted as positive. $Precison = \frac{TP}{TP+FP}$

Parameter	Value
Batch size	1000
Number of epochs	250
Path length	500
Gradient steps per epoch	100
Discount factor	0.99
Optimizer	Adam
Policy hidden sizes	(256, 256, 256)
Policy activation function	ReLU
Policy learning rate	3×10^{-4}
Policy minimum standard deviation	e^{-20}
Policy maximum standard deviation	e^{2}
Soft target interpolation	5×10^{-3}
FedT-DQN	
Transformer layers	2
Transformer heads	4
Transformer hidden sizes	256
Encoder hidden size	(256, 256, 256)
Decoder hidden size	(256, 256, 256)
Activation function	ReLU
Q-function learning rate	3×10^{-4}
SAC	
Hidden size	(256, 256, 256)
Activation function	ReLU
Q-function learning rate	3×10^{-4}

Table 4. Model parameters configuration.

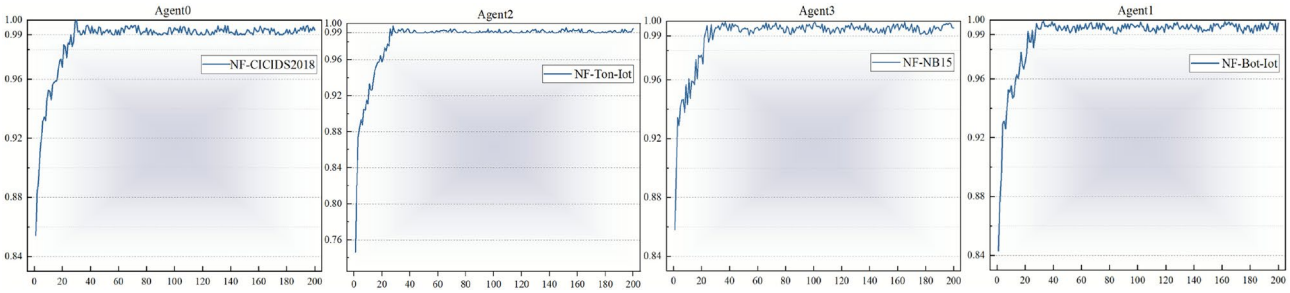

Fig. 2. Changes in Accuracy over 200 rounds. The x-axis represents the rounds, and the y-axis represents the accuracy.

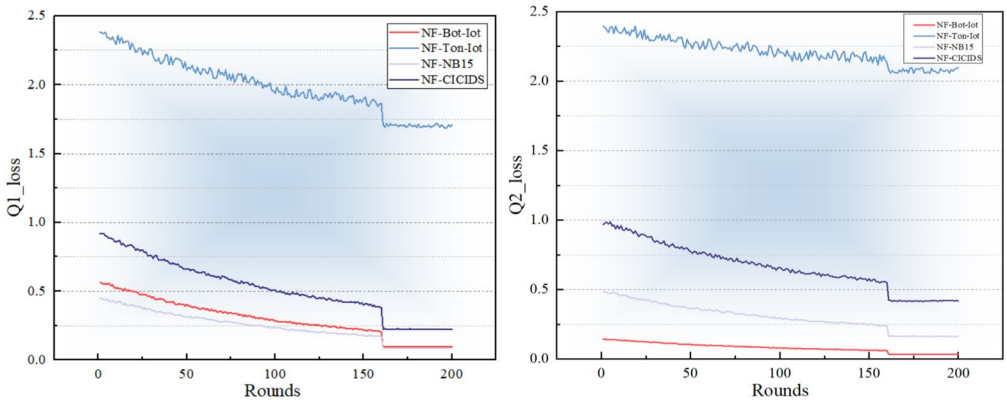

Fig. 3. Changes in q1_loss and q2_loss on four clients during the 200 rounds of training.

- **Recall**: It is the ratio of the number of correctly predicted positive predictions to the number of samples that are actually positive. $Recall = \frac{TP}{TP+FN}$
- **F1-Score**: It is the harmonic mean of recall and precision. $F1 - Score = \frac{2 \times Precison \times Recall}{Precison + Recall}$
- **Area Under the Curve (AUC)**: is the area under the curve bordered by two parameters: true positive rate (TPR) and false positive rate (FPR). $TPR = \frac{TP}{TP+FN}$ $FPR = \frac{FP}{FP+TN}$

Experimental results

Performance comparison across different datasets

We evaluated the performance of our model on the aforementioned four datasets. For all experiments, we plotted the model's accuracy and the achieved loss over the 200 training rounds, as shown in Figs. 2 and 3. Here, Agents 0 to 3 were trained on "NF-cicids2018-v2.csv", "NF-bot-iot-v2.csv", "NF-ton-iot-v2.csv", and "NF-nb15-v2.csv", respectively, in a parallel manner. The best Accuracy achieved was 0.9922, 0.9987, 0.9984, and 0.9957, respectively. Comparing Q1_loss and Q2_loss, we observed that on the NF-cicids2018-v2, NF-bot-iot-v2, and NF-nb15-v2 datasets, the Q1 network exhibited smaller learning errors in its state-action value function, whereas the NF-bot-iot-v2 dataset showed lower loss on the Q2 network. To the best of our knowledge, this is the first federated learning work on intrusion detection research that achieves such high performance simultaneously on all four datasets.In prior state-of-the-art research, the TS-IDS[53] model achieved the best performance on these four datasets for the binary classification task. Specifically, it attained the highest accuracy scores of 0.9843, 0.8821, 0.9985, and 0.9952 on the NF-CSE-CIC-IDS2018-v2, NF-BoT-IoT-v2, NF-ToN-IoT-v2, and NF-UNSW-NB15-v2 datasets, respectively. Our proposed FedT-DQN model, however, outperforms these results.

we evaluated the average of various metrics on each agent client as well as the average of the metrics of the global model, as shown in Fig. 4.

Comparison with other baselines

The Table 5 shows the binary classification results (ACC, AUC, macro precision, macro recall, and weighted F1) of our proposed model and the baselines on the four benchmark datasets.We have highlighted the data with the best performance in bold. We have the following findings: (1) Our model FedT-DQN performs exceptionally well on the baseline datasets compared to other federated learning models.(2) Compared to FTKD model[54], our proposed method achieves better results on the four datasets. The Accuracy for NF-BoT-IoT-v2, NF-ToN-IoT-v2, NF-UNSW-NB15-v2, and NF-CICIDS2018-v2 increased by 5.36%, 2.12%, 3.91%, and 5.53%, respectively. This demonstrates the benefits of incorporating the SAC policy network and self-attention dynamic aggregation

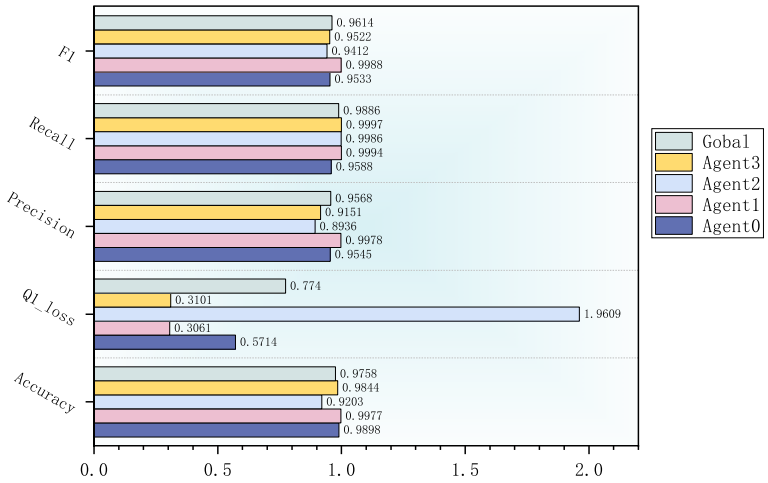

Fig. 4. The average performance metrics under different clients and the average performance value of the global model.

Dataset	Metric	FELIDS[44]	FTKD[54]	FDRL-IDS[46]	FCL-SBLS[55]	FedT-DQN
NF-BoT-IoT-v2	ACC	0.9369	0.9451	0.9209	0.944	**0.9987**
	AUC	0.9256	0.9044	0.9375	0.915	**0.9936**
	Recall	0.9461	0.9477	0.9219	0.943	**0.9951**
	Precision	0.9133	0.9114	0.9105	0.948	**0.9978**
	F1-Score	0.9224	0.9515	0.9397	0.946	**0.9993**
NF-ToN-IoT-v2	ACC	0.9011	0.9772	0.9309	0.8953	**0.9984**
	AUC	0.8924	0.9715	0.9567	0.8653	**0.9936**
	Recall	0.8861	0.9772	0.9677	0.9017	**0.9994**
	Precision	0.8133	0.9111	0.9325	0.8541	**0.9384**
	F1-Score	0.9014	0.9771	0.9577	0.8842	**0.9622**
NF-UNSW-NB15-v2	ACC	0.9145	0.9566	0.9609	0.9234	**0.9957**
	AUC	0.9556	0.9055	0.9778	0.9341	**0.9992**
	Recall	0.9436	0.8955	0.9514	0.9414	**0.9921**
	Precision	0.9445	0.9157	0.9384	0.9019	**0.9551**
	F1-Score	0.9140	0.9255	0.9325	0.9287	**0.9522**
NF-CICIDS2018-v2	ACC	0.9633	0.9369	0.9144	0.9087	**0.9922**
	AUC	0.9644	0.9511	0.9079	0.8723	**0.9942**
	Recall	0.9556	0.9869	0.9655	0.9236	0.9564
	Precision	0.9521	0.9611	0.9445	0.9187	**0.9679**
	F1-Score	0.9192	0.9864	0.9145	0.9335	**0.9594**

Table 5. Performance comparison of federated learning models across four datasets. Significant values are in bold.

in federated learning for IDS. (3) Compared to the FDRL-IDS[46] learning method, we found that all metrics achieved superior results. We believe that compared to the FDRL-IDS method, our inclusion of the Transformer encoder is beneficial for aggregating client knowledge. The attention heads focus on specific types of knowledge patterns, which significantly reduces misclassification.(4)In comparison with FCL-SBLS[55], FedT-DQN boosts detection accuracy by 5 to 9 percentage points by enhancing its deep modeling capability and resolving the core issue of heterogeneous data fusion at the aggregation mechanism level. This approach remedies its notable deficiencies in both detection accuracy and cross-dataset generalization.

On the UNSW-MG24 and CIC IIoT 2025 datasets, we compared FedT-DQN with FDRL-IDS,FTKD and FCL-SBLS, three models geared towards IoT scenarios. As shown in Table 6, FedT-DQN consistently outperformed the other models across all metrics. The key difference lies in FedT-DQN's use of the SAC network for policy optimization and its superior contextual modeling capability. This demonstrates its excellent ability to handle traffic data across diverse scenarios.

Dataset	Metric	FDRL-IDS[46]	FCL-SBLS[55]	FTKD[54]	FedT-DQN
UNSW-MG24	ACC	0.8571	0.9141	0.9255	**0.9517**
	AUC	0.8415	0.8851	0.9111	**0.9114**
	Recall	0.9124	0.9045	0.9125	**0.9536**
	Precision	0.9241	0.9167	0.9633	**0.9645**
	F1	0.9057	0.9221	0.9721	**0.9765**
CIC IIoT 2025	ACC	0.8438	0.8944	0.9511	**0.9678**
	AUC	0.8566	0.9124	0.9111	**0.9211**
	Recall	0.8745	0.9055	0.9005	**0.9577**
	Precision	0.8612	0.8766	0.8911	**0.9155**
	F1	0.8477	0.8748	0.9251	**0.9366**

Table 6. Performance comparison of federated learning models in UNSW-MG24 and CIC IIOT 2025. Significant values are in bold.

Dataset	ACC	AUC	Prec.	Rec.	F1
Random distribution					
CICIDS2018-v2	0.9945	0.9918	0.9811	0.9781	0.9987
BoT-IoT-v2	0.9901	0.9945	0.9841	0.9838	0.9936
ToN-IoT-v2	0.9871	0.9941	0.9414	0.9814	0.9710
UNSW-NB15-v2	0.9881	0.9945	0.9488	0.9996	0.9774
Customized distribution					
CICIDS2018-v2	0.9922	0.9942	0.9679	0.9564	0.9594
BoT-IoT-v2	0.9987	0.9936	0.9978	0.9951	0.9993
ToN-IoT-v2	0.9984	0.9936	0.9384	0.9994	0.9622
UNSW-NB15-v2	0.9957	0.9992	0.9551	0.9921	0.9522

Table 7. Performance comparison of FedT-DQN under heterogeneous data distribution scenarios.

Aggregation type	Dataset	ACC	AUC	Prec.	Rec.	F1
FedT-DQN	NF-CICIDS2018-v2	0.9945	0.9918	0.9811	0.9781	0.9987
	NF-BoT-IoT-v2	0.9901	0.9945	0.9841	0.9838	0.9936
	NF-ToN-IoT-v2	0.9871	0.9941	0.9414	0.9814	0.9710
	NF-UNSW-NB15-v2	0.9881	0.9945	0.9488	0.9996	0.9774
FedT-DQN-Avg	NF-CICIDS2018-v2	0.9764	0.9522	0.9600	0.9664	0.9221
	NF-BoT-IoT-v2	0.9881	0.9667	0.9224	0.9743	0.9891
	NF-ToN-IoT-v2	0.9512	0.9211	0.9084	0.9134	0.9335
	NF-UNSW-NB15-v2	0.9422	0.9316	0.9077	0.9474	0.9091

Table 8. Ablation study on the aggregation method, comparing the performance impact of including versus excluding the Transformer's self-attention aggregation.

Processing results for randomly distributed data

Table 7 presents the evaluation results of our secondary experimental configuration, in which the entire dataset was randomly distributed evenly among participating agents. The results reveal remarkably consistent performance metrics between randomly partitioned data scenarios and customized distribution schemes, confirming our framework's robust capability in managing non-independently and identically distributed (non-iid) data environments.

Ablation study

The impact of the aggregation algorithm

In this section, we evaluate a variant of our method, FedT-DQN-avg, where we replace the aggregation method with FedAvg and analyze it in the four datasets with a random distribution. The results of the ablation analysis are shown in Table 8, which shows that FedT-DQN outperforms FedT-DQN-avg, proving that after incorporating the policy network, dynamic self-attention aggregation is superior to FedAvg.

With/Without SAC	Dataset	ACC	AUC	Prec.	Rec.	textbfF1
FedT-DQN	NF-CICIDS2018-v2	0.9945	0.9918	0.9811	0.9781	0.9987
	NF-BoT-IoT-v2	0.9901	0.9945	0.9841	0.9838	0.9936
	NF-ToN-IoT-v2	0.9871	0.9941	0.9414	0.9814	0.9710
	NF-UNSW-NB15-v2	0.9881	0.9945	0.9488	0.9996	0.9774
FedT-DQN-Non SAC	NF-CICIDS2018-v2	0.9244	0.9157	0.9487	0.9024	0.9002
	NF-BoT-IoT-v2	0.8915	0.9156	0.9355	0.9051	0.9361
	NF-ToN-IoT-v2	0.9166	0.9238	0.9081	0.9552	0.9315
	NF-UNSW-NB15-v2	0.9211	0.9361	0.9126	0.9235	0.9022

Table 9. Ablation study on the aggregation method, comparing the performance impact of including versus excluding the SAC network.

Client	Running time (s)
Agent 0 (NF-CICIDS2018-v2)	0.6315
Agent 1 (NF-Bot-IoT-v2)	0.7441
Agent 2 (NF-ToN-IoT-v2)	0.5691
Agent 3 (NF-NB15-v2)	0.8642
Average	0.7022

Table 10. Running time (second) of FedT-DQN per batch.

The impact of the SAC network

We also discuss the impact of the SAC network on the model. In Table 9, we compare the performance on metrics such as F1-Score and AUC with and without the SAC network. It can be observed that the FedT-DQN model incorporating the SAC network demonstrates significantly higher performance. This indicates that the integration of SAC not only improves the stability of decision-making but also enhances the model's adaptability in dynamic environments. By introducing entropy regularization, SAC effectively balances exploration and exploitation, thereby optimizing the global defense strategy.

Running time

In federated learning, distributed node overhead is an unavoidable consideration. We report the actual running time of our proposed method per epoch, as our approach performs batch processing for both training and inference phases. We observed that the running times of these two phases are similar; therefore, we only present the per-batch running time of the training phase in the Table 10.

Computer complexity

In this section, we explore the time complexity of FedT-DQN and several other federated reinforcement learning models. The authors of the other models discussed herein did not provide calculations for their time complexity. Therefore, we present the time complexity of each model based on its components for the reader's reference.

Let K denote the number of federated clients,n_k the number of local samples on client k,d the input feature dimension (d= 43 NetFlow attributes). H the hidden size of the encoder and Q-networks (H= 256). B the batch size and G the number of gradient steps per round. Each client maintains one policy network and two Q-networks implemented as three-layer MLPs. For a single MLP, the forward and backward pass on one sample requires $O(dH + H^2)$ operations, where the term H^2 dominates in our setting. Therefore, one gradient update on a mini-batch of size B incurs $O(BH^2)$ complexity per client Summing over G updates, R communication rounds and K clients, the overall local SAC training cost is:

$$T_{(SAC)} = O(RKGBH^2) \tag{8}$$

In FedT-DQN, the federated aggregation is implemented by a Transformer encoder that processess equence of length $K + 1$ (one embedding per client plus a CLS token). For each Transformer layer with hidden size H, computing the multi-head self-attention and the feed-forward network requires $O(S^2H + SH^2)$ operations, where $S = K + 1$ is the sequence length. Hence, with L layers the aggregation cost is

$$T_{(Trans)} = O(L(K^2H + KH^2)) \tag{9}$$

As K is typically small (e.g., K= 4 in our experiments),this term has the same order as a single forward pass of the local networks and does not change the overall asymptotic complexity, which remains dominated by T_{sac}.We compared FedT-DQN with other models in Table 11.

FELIDS: The computational complexity of FELIDS consists of three main parts, local training, global aggregation, and parameter exchange. The complexity of local training is $O(E \cdot nk \cdot (l_0 \cdot l_1 + \cdots + l_L \cdot l_{L+1}))$, where E is

Model	Complexity
FELIDS	$O(FELIDS) = O(K \cdot E \cdot n_k \cdot (l_0 \cdot l_1 + \cdots + l_L \cdot l_{L+1})) + O(K \cdot W) + O(W)$
FCL-SBLS	$O(I \cdot D_i \cdot \lvert w_i \rvert \cdot Q + \lvert w_g \rvert \cdot f_p \cdot N_r)$
DRL-IDS	$O(R \cdot G \cdot S \cdot A \cdot N)$
FTKD	$O(N^2)$
FedT-DQN	$O(RKGBH^2)$

Table 11. Comparison of computational complexity between models.

the number of local training epochs, n_k is the local data size for client *k*, l_x is the number of nodes in layer *x*, and *L* is the number of layers in the model. The complexity of global aggregation is $O(K \cdot W)$, where *K* is the number of clients and *W* is the number of model parameters per client. The complexity of parameter exchange is *O*(*W*). The overall complexity is:$O(FELIDS) = O(K \cdot E \cdot n_k \cdot (l_0 \cdot l_1 + \cdots + l_L \cdot l_{L+1})) + O(K \cdot W) + O(W)$

FCL-SBLS: FCL-SBLS's complexity comes from local training and global aggregation. Local training for each UAV has complexity $|D_i| \cdot O(|w_i| \cdot f_p \cdot Q)$, and global aggregation is $O(|w_g| \cdot f_p)$, where D_i is the local dataset size, $|w_i|$ is the number of model weights, and *Q* is the number of SBLS blocks.The overall complexity is $O(I \cdot D_i \cdot |w_i| \cdot Q + |w_g| \cdot f_p \cdot N_r)$, where *I* is the number of UAVs, and N_r is the number of training epochs.

DRL-IDS: DRL-IDS using PPO2 has complexity $O(G \cdot S \cdot A \cdot N)$, where *G* is the number of gradient steps, *S* is the state space size, *A* is the action space size, and *N* is the batch size. Over *R* rounds of training, the overall complexity is $O(R \cdot G \cdot S \cdot A \cdot N)$

FTKD: The time complexity of FTKD is dominated by local model training and EMMD aggregation. Local training has a complexity of $O(N \cdot D)$, and the EMMD aggregation, which involves pairwise calculations, has a complexity of $O(N^2)$. Thus, the overall time complexity is $O(N^2)$. Making it efficient for moderate data sets but potentially slower for large ones.

Discussion

The federated learning framework adopted in this study protects the sensitive data of clients through local data storage and model aggregation. In federated learning, data is always kept locally, and only encrypted model updates are transmitted to a central server, thereby effectively preventing the leakage of raw data. To further enhance privacy protection, this model can also incorporate differential privacy techniques by adding noise to the model updates, ensuring that clients' original data cannot be reconstructed during parameter aggregation.

However, despite its significant advantages in privacy protection, federated learning still has potential limitations and attack surfaces. First, malicious clients may compromise the quality of the global model or even leak other clients' sensitive information by uploading manipulated model updates, Such attacks can infer sensitive data by analyzing model updates (e.g., gradient information). Therefore, preventing attacks from malicious clients while ensuring privacy remains a challenge in federated learning frameworks, Furthermore, the communication overhead and bandwidth limitations of federated learning can also affect the effectiveness of privacy protection, especially in edge computing environments, Balancing privacy protection with computational efficiency is a key direction for future research.

Conclusion and future directions

The FedT-DQN framework proposed in this study effectively addresses key challenges faced by network intrusion detection systems in IoT environments, including privacy protection, dynamic adaptability, and cross-domain generalization. It is the first to introduce a Transformer encoder into the federated aggregation process, achieving context-aware dynamic knowledge fusion through self-attention mechanisms, thereby overcoming the limitations of traditional FedAvg parameter averaging. The designed dual-layer Q-network architecture effectively separates feature learning and policy optimization processes, and combined with a dynamic client selection algorithm, significantly enhances detection performance in heterogeneous network environments. Experimental results demonstrate that FedT-DQN achieves significant performance improvements compared to traditional federated learning methods on four benchmark datasets. Additionally, in terms of aggregation algorithms, we propose dynamic aggregation based on self-attention mechanisms, providing a viable technical path for multi-institutional cybersecurity collaboration.

Although the FedT-DQN framework has shown promising results, it still faces challenges in large-scale deployment, including the high computational complexity of the Transformer attention mechanism, substantial communication overhead, insufficient robustness against adversarial attacks, and limited adaptability to concept drift. Future research will focus on the following directions: exploring efficient variants such as linear attention and sparse attention to reduce computational complexity; researching model compression and gradient quantization techniques to optimize communication efficiency; designing Byzantine-robust mechanisms to counter malicious client attacks; developing continual learning mechanisms to address evolving attack patterns; integrating multimodal security data sources to enhance detection comprehensiveness; and validating the practicality and scalability of the framework in real large-scale IoT environments. These research directions will drive the development of intelligent intrusion detection technologies in federated environments toward greater maturity and practicality.

Data availability

The Netflow datasets used in this study, "NF-BoT-IoT, NF-ToN-IoT, NF-CSE-CIC-IDS2018-v2, and NF-UNSW-NB15-v2," are publicly available at https://staff.itee.uq.edu.au/marius/NIDS_datasets/. "UNSW-MG24,CIC IIOT 2025"are publicly available at https://ieee-dataport.org/documents/unsw-mg24 and https://www.unb.ca/cic/datasets/iiot-dataset-2025.html

Received: 5 October 2025; Accepted: 4 December 2025
Published online: 12 December 2025

References

1. Aslan, Ö., Aktuğ, S. S., Ozkan-Okay, M., Yilmaz, A. A. & Akin, E. A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions. *Electronics* **12**, 1333 (2023).
2. Lv, Z., Han, Y., Singh, A. K., Manogaran, G. & Lv, H. Trustworthiness in industrial IoT systems based on artificial intelligence. *IEEE Trans. Industr. Inf.* **17**, 1496–1504 (2020).
3. Bekri, W., Jmal, R. & Fourati, L. C. Secure and trustworthiness IoT systems: investigations and literature review. *Telecommun. Syst.* **85**, 503–538 (2024).
4. Zuech, R., Khoshgoftaar, T. M. & Wald, R. Intrusion detection and big heterogeneous data: a survey. *J. Big Data* **2**, 1–41 (2015).
5. Khan, M. A. & Kim, J. Toward developing efficient conv-ae-based intrusion detection system using heterogeneous dataset. *Electronics* **9**, 1771 (2020).
6. Kuang, F., Xu, W. & Zhang, S. A novel hybrid kpca and svm with ga model for intrusion detection. *Appl. Soft Comput.* **18**, 178–184 (2014).
7. Reddy, R. R., Ramadevi, Y. & Sunitha, K. N. Effective discriminant function for intrusion detection using svm, in *2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 1148–1153 (IEEE, 2016).
8. Farnaaz, N. & Jabbar, M. Random forest modeling for network intrusion detection system. *Procedia Comput. Sci.* **89**, 213–217 (2016).
9. Li, Z., Liu, F., Yang, W., Peng, S. & Zhou, J. A survey of convolutional neural networks: Analysis, applications, and prospects. *IEEE Trans. Neural Netw. Learn. Syst.* **33**, 6999–7019 (2021).
10. Dina, A. S. & Manivannan, D. Intrusion detection based on machine learning techniques in computer networks. *Internet Things* **16**, 100462 (2021).
11. Tsai, C.-F., Hsu, Y.-F., Lin, C.-Y. & Lin, W.-Y. Intrusion detection by machine learning: A review. *Expert Syst. Appl.* **36**, 11994–12000 (2009).
12. Lansky, J. et al. Deep learning-based intrusion detection systems: A systematic review. *IEEE Access* **9**, 101574–101599 (2021).
13. Ahmed, S. F. et al. Deep learning modelling techniques: Current progress, applications, advantages, and challenges. *Artif. Intell. Rev.* **56**, 13521–13617 (2023).
14. Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M. & Bouchachia, A. A survey on concept drift adaptation. *ACM Comput. Surveys (CSUR)* **46**, 1–37 (2014).
15. Kumar, V., Sinha, D., Das, A. K., Pandey, S. C. & Goswami, R. T. An integrated rule based intrusion detection system: analysis on unsw-nb15 data set and the real time online dataset. *Clust. Comput.* **23**, 1397–1418 (2020).
16. Stojkoska, B. L. R. & Trivodaliev, K. V. A review of internet of things for smart home: Challenges and solutions. *J. Clean. Prod.* **140**, 1454–1464 (2017).
17. Yurtsever, E., Lambert, J., Carballo, A. & Takeda, K. A survey of autonomous driving: Common practices and emerging technologies. *IEEE Access* **8**, 58443–58469 (2020).
18. McMahan, B., Moore, E., Ramage, D., Hampson, S. & y Arcas, B. A. Communication-efficient learning of deep networks from decentralized data, in *Artificial intelligence and statistics*, 1273–1282 (PMLR, 2017).
19. Wang, J., Liu, Q., Liang, H., Joshi, G. & Poor, H. V. Tackling the objective inconsistency problem in heterogeneous federated optimization. *Adv. Neural. Inf. Process. Syst.* **33**, 7611–7623 (2020).
20. Watkins, C. J. & Dayan, P. Q-learning. *Mach. Learn.* **8**, 279–292 (1992).
21. Li, W., Yi, P., Wu, Y., Pan, L. & Li, J. A new intrusion detection system based on KNN classification algorithm in wireless sensor network. *J. Electric. Comput. Eng.* **2014**, 240217 (2014).
22. Quinlan, J. R. Induction of decision trees. *Mach. Learn.* **1**, 81–106 (1986).
23. Jain, A. K. & Dubes, R. C. *Algorithms for Clustering Data* (Prentice-Hall, Inc., 1988).
24. Saxena, A. et al. A review of clustering techniques and developments. *Neurocomputing* **267**, 664–681 (2017).
25. Bivens, A. et al. Network-based intrusion detection using neural networks. *Intell. Eng. Syst. Artif. Neural Netw.* **12**, 579–584 (2002).
26. Aljawarneh, S., Aldwairi, M. & Yassein, M. B. Anomaly-based intrusion detection system through feature selection analysis and building hybrid efficient model. *J. Comput. Sci.* **25**, 152–160 (2018).
27. Muhammad, G., Hossain, M. S. & Garg, S. Stacked autoencoder-based intrusion detection system to combat financial fraudulent. *IEEE Internet Things J.* **10**, 2071–2078 (2020).
28. Siddiqui, A. J. & Boukerche, A. Adaptive ensembles of autoencoders for unsupervised IoT network intrusion detection. *Computing* **103**, 1209–1232 (2021).
29. Karaca, K. N. & Çetin, A. Botnet attack detection using convolutional neural networks in the IoT environment, in *2021 International Conference on Innovations in Intelligent Systems and Applications (INISTA)*, 1–6 (IEEE, 2021).
30. Le, T.-T.-H., Kim, Y. & Kim, H. Network intrusion detection based on novel feature selection model and various recurrent neural networks. *Appl. Sci.* **9**, 1392 (2019).
31. Ullah, I. & Mahmoud, Q. H. Design and development of RNN anomaly detection model for IoT networks. *IEEE Access* **10**, 62722–62750 (2022).
32. Wang, H. & Li, W. Ddostc: A transformer-based network attack detection hybrid mechanism in SDN. *Sensors* **21**, 5047 (2021).
33. Qureshi, A.-u.-H., Larijani, H., Ahmad, J. & Mtetwa, N. A heuristic intrusion detection system for internet-of-things (iot), in *Intelligent Computing: Proceedings of the 2019 Computing Conference*, Vol. 1, 86–98 (Springer, 2019).
34. Hassan, M. M., Gumaei, A., Alsanad, A., Alrubaian, M. & Fortino, G. A hybrid deep learning model for efficient intrusion detection in big data environment. *Inf. Sci.* **513**, 386–396 (2020).
35. Koca, M., Aydin, M. A., Sertbaş, A. & Zaim, A. H. A new distributed anomaly detection approach for log ids management based ondeep learning. *Turk. J. Electr. Eng. Comput. Sci.* **29**, 2486–2501 (2021).
36. AVCI, İ., Koca, M. & Atasoy, M. Windows tabanlı uygulamalarda sql enjeksiyon siber saldırı senaryosu ve güvenlik önlemleri. avrupa bilim ve teknoloji dergisi (28), 213–219 (2021).
37. Jing, D. & Chen, H. B. Svm based network intrusion detection for the unsw-nb15 dataset, in *2019 IEEE 13th International Conference on ASIC (ASICON)* (2019).
38. Ingre, B., Yadav, A. & Soni, A. K. *Decision Tree Based Intrusion Detection System for nsl-kdd Dataset* (Springer, 2017).
39. Pacheco, J., Benitez, V. H., Filix-Herran, L. C. & Satam, P. Artificial neural networks based intrusion detection system for internet of things fog nodes. *IEEE Access* 1 (2020).

40. Meerja, A. J., Ashu, A. & Rajani Kanth, A. Gaussian naïve bayes based intrusion detection system, in *Proceedings of the 11th International Conference on Soft Computing and Pattern Recognition (SoCPaR 2019)*, Vol. 11, 150–156 (Springer, 2021).
41. Chen, L., Kuang, X., Xu, A., Suo, S. & Yang, Y. A novel network intrusion detection system based on CNN, in *2020 Eighth International Conference on Advanced Cloud and Big Data (CBD)*, 243–247 (IEEE, 2020).
42. Nayyar, S., Arora, S. & Singh, M. Recurrent neural network based intrusion detection system, in *2020 International Conference on Communication and Signal Processing (iccsp)*, 0136–0140 (IEEE, 2020).
43. de Carvalho Bertoli, G., Junior, L. A. P., Saotome, O. & Dos Santos, A. L. Generalizing intrusion detection for heterogeneous networks: A stacked-unsupervised federated learning approach. *Comput. Security* **127**, 103106 (2023).
44. Friha, O. et al. Felids: Federated learning-based intrusion detection system for agricultural internet of things. *J. Parallel Distrib. Comput.* **165**, 17–31 (2022).
45. Ruzafa-Alcázar, P. et al. Intrusion detection based on privacy-preserving federated learning for the industrial IoT. *IEEE Trans. Industr. Inf.* **19**, 1145–1154 (2021).
46. Vadigi, S., Sethi, K., Mohanty, D., Das, S. P. & Bera, P. Federated reinforcement learning based intrusion detection system using dynamic attention mechanism. *J. Inf. Security Appl.* **78**, 103608 (2023).
47. Chen, X., Qiu, W., Chen, L., Ma, Y. & Ma, J. Fast and practical intrusion detection system based on federated learning for vanet. *Comput. Security* **142**, 103881 (2024).
48. Bellman, R. A Markovian decision process. *J. Math. Mech.* 679–684 (1957).
49. Haarnoja, T., Zhou, A., Abbeel, P. & Levine, S. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, in *International Conference on Machine Learning*, 1861–1870 (PMLR, 2018).
50. Hebert, L., Golab, L., Poupart, P. & Cohen, R. Fedformer: Contextual federation with attention in reinforcement learning. arXiv preprint arXiv:2205.13697 (2022).
51. Mnih, V. et al. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602 (2013).
52. Schaul, T., Quan, J., Antonoglou, I. & Silver, D. Prioritized experience replay. arXiv preprint arXiv:1511.05952 (2015).
53. Nguyen, H. & Kashef, R. Ts-ids: traffic-aware self-supervised learning for IoT network intrusion detection. *Knowl.-Based Syst.* **279**, 110966 (2023).
54. Zhou, R., Wang, Z., Yang, S., He, D. & Chan, S. Federated learning based on two-stage knowledge distillation for intrusion detection in industrial IoT. *Expert Syst. Appl.* 130144 (2025).
55. He, X. et al. Federated continuous learning based on stacked broad learning system assisted by digital twin networks: An incremental learning approach for intrusion detection in UAV networks. *IEEE Internet Things J.* **10**, 19825–19838 (2023).

Author contributions

Author contribution: Conceptualization, T.M. and W.Y.; methodology, T.M. and S.W.; validation, Y.Y.; investigation, Y.Y. and S.W.; resources, W.Y. and T.M.; writing—original draft preparation, T.M.; writing—review and editing, T.M.

Funding

This research was funded by the Key Research and Development Program of the Autonomous Region (Grant No.2022B01008), the National Natural Science Foundation of China (Grant No.62262065), the "Tianshan Talent" Research Project of Xinjiang (Grant No.2022TSYCLJ0037).

Declarations

Competing interests

The authors declare no conflicts of interest.

Additional information

Correspondence and requests for materials should be addressed to Y.Y. or W.Y.