RedPO-BRNNet: Federated Anomaly Detection with Differential Privacy and Secure Aggregation for Edge IoT

Rolly R. Tang
Digital Innovation Research Center, Xi'an Mingde Institute of Technology, Xi'an, Shaanxi, 710124, China
E-mail: tangrolly2019@yeah.net

Keywords: Federated learning (FL), predictive performance, model training, red panda optimizer fused bidirectional recurrent neural network (RedPO-BRNNet), local data

Received: August 27, 2025

In the context of edge computing (EC), there are distributed Internet of Things (IoT) devices that generate huge volumes of sensitive data, which necessitates privacy-preserving and efficient model training. Federated learning (FL) enables the ability to collaboratively train models without sharing raw data, albeit at the risk of sharing data values through indirect means and in an uncontrolled manner that continues to expose privacy risks like gradient inference attacks and data reconstruction methods. To propose a secure aggregation framework that combines FL with differential privacy (DP) at the 'noise' level of 1.e−4 and a red panda optimizer fused bidirectional recurrent neural network (RedPO-BRNNet), that improves predictive performance, convergence efficiency, and robustness in an EC environment. This framework is implemented using Python and utilizes local datasets of 2,501 time-series recordings gathered from 10 distributed edge devices, which feature blood pressure, activity level, body temperature, heart rate, anomaly scores, and a binary classification label. The data are pre-processed using Z-score normalization to standardize scales for the features across devices. Each edge device trains a local RedPO-BRNNet model, applies DP noise to the model parameters and then contributes updates that preserve privacy to the global aggregation using FedAvg. RedPO-BRNNet is tested against LSTM, Bi-LSTM, and baseline BRNN models during five training epochs. The proposed model achieves 94.97% anomaly detection accuracy, 2.14% privacy leakage, 57.82 ms client latency, and 78.96% model convergence efficiency, with improvements statistically validated using paired t-tests (p < 0.05). These results show that RedPO-BRNNet effectively protects privacy while enabling efficient and scalable edge intelligence, making it a dependable solution for safe, multi-device IoT deployments.

Povzetek:

1 Introduction

Edge computing was utilized to efficiently manage the vast amount of data generated by various smart devices, particularly in the era of the rapidly growing Internet of Things (IoT) [1]. Unlike cloud computing, which is based on centralized servers, edge computing analyzes data locally, closer to the source, lowering latency, increasing decision-making speed, and optimizing bandwidth utilization [2]. However, as data processing shifts to distributed edge nodes, new issues arise in maintaining privacy, assuring security, and conducting effective data aggregation [3]. In real-world applications such as autonomous vehicles, healthcare monitoring, and smart environments, transferring raw data to central servers raises privacy concerns, necessitating the use of privacy-preserving federated and decentralized learning systems [4]. Federated Learning (FL) evolved as a viable paradigm to solve these difficulties by allowing devices to train local models on their own data and only share model updates (weights or gradients) with a central server, rather than raw data [5]. This decentralized technique ensured user anonymity while reducing communication overhead. However, investigations have demonstrated that even model gradients can be used to reconstitute private data [6]. Differential Privacy (DP) was introduced into FL to improve privacy by adding controlled noise to common parameters. This guaranteed that changes in individual data points had a low impact on the global model output [7], hence improving dependability and secrecy in privacy-preserving learning frameworks [8]. Due to the limited internet connectivity, low power, and short battery life of edge devices, applying Federated Learning (FL) and Differential Privacy (DP) for secure data aggregation in edge computing presented considerable hurdles [9]. As a result, any secure aggregation method in such situations must be scalable, lightweight, and resistant to malicious attacks. Furthermore, ensuring data integrity and accuracy while avoiding privacy intrusions is critical [10]. Integrating FL and DP into edge computing necessitated a well-defined framework that balanced resource efficiency with robust and secure data aggregation [11]. Because edge devices frequently encounter energy and bandwidth limits, the aggregation mechanism must be fault-tolerant and adaptable to client dropouts [12].

This required efficient communication protocols and adaptive learning models to handle noisy or missing inputs from edge nodes, alongside the development of secure aggregation mechanisms to counter both passive and active adversarial attacks [13]. Several limitations were identified, including challenges in achieving scalability with real-time performance on limited resources, substantial computational demands on edge nodes, vulnerability to advanced attack vectors, and potential trade-offs between model accuracy and privacy. To improve predictive accuracy, data confidentiality, and computational efficiency in distributed edge computing environments by combining FL with DP to create a safe and private data aggregation framework.The main contributions are as follows:

Privacy-Preserving Federated Learning Framework: To safeguard user data in edge computing settings, a secure model aggregation technique that combines Federated Learning (FL) and Differential Privacy (DP) was introduced.

Robust Dataset Collection and Preprocessing: To guarantee uniform feature scaling across all edge nodes, Z-score normalisation was used in conjunction with real-time time-series data from dispersed IoT devices.

Effective Hybrid Model Design: To increase anomaly detection precision and accelerate convergence in federated environments, a Red Panda Optimizer-fused Bidirectional Recurrent Neural Network (RedPO-BRNNet) was created.

2 Literature review

FL-RAEC utilized phased aggregation and hybrid privacy methods to protect information [14]. It displayed good resilience to malicious interference with high computational costs and initial trust authentication requirements, emphasizing the importance of secure and expandable aggregation in decentralized AI learning. An adaptive gradient compression and differential privacy-empowered hybrid federated edge learning system augmented industrial data privacy and defeated inference attacks [15]. Despite additional complexity and processing cost, it reduced latency considerably and improved real-time security, with a focus on scalable privacy-preserving learning in decentralized networks. Blockchain FL incorporated distributed K-means, random forest, and AdaBoost models combined with homomorphic encryption and differential privacy to secure IIoT data aggregation [16]. Experiments demonstrated improved accuracy and privacy but latency and computing workload remained an issue. It delivered greater protection from model extraction and reverse engineering, highlighting the importance of scalable, crypto-based FL aggregation.

In an effort to counter privacy attacks and leakage across edge networks, a Differential Privacy-Fuzzy Convolution Neural Network (DP-FCNN) was designed, which combines the Laplace mechanism and a Fuzzy CNN [17]. With Java implementation and the use of Merkle trees, BLAKE2 hashing, and Piccolo encryption, it enhanced accuracy, speed of processing, and scalability but limited real-time flexibility and resilience.

A light scheme named Privacy Protection FL for Edge Computing (PPFLEC) was put forward to defend privacy in edge computing through hash functions, digital signatures, and weight masking to assist in secret sharing with FL [18].The method facilitated data integrity, gradient protection, and replay and collusion attack resilience. Comparison results showed that differential privacy performed 40% worse. However, the method was challenged by constraints in edge instability and low scalability. Data privacy and integrity were prioritized under limited computational capabilities within Internet of Medical Things (IoMT) situations. Table 1 displays the literature review of existing methods.

2.1 Problem statement

Federated learning solutions [19-29] show great progress in the ability to perform privacy-preserving computation via approaches like differential privacy, homomorphic encryption, and using blockchain. Still, existing models struggle with scalability when there are client dropouts [21], [27], computational overhead [22], [28], dependence on hardware-specific strategies [26], and little evaluation of gradient inversion or real-world settings with heterogeneity [19], [23]. For example, most models that have been proposed do not measure resiliency to network effects, such as latency and packet loss. The model proposed in this paper, RedPO-BRNNet, addresses these limitations by integrating Red Panda Optimization into Differential Privacy to achieve higher accuracy on the model, faster convergence, and very strong resistance to gradient leakage when compared to realistic edge-network environments.

3 Methodological framework

The Secure Edge IoT dataset, based on discrete device biometric data, is used for safe anomaly detection in edge-based IoT networks. Standardization of feature scales and local hybrid RedPO-BRNNet model training ensures differential privacy. Figure 1 illustrates the complete process of the Secure Data Aggregation Scheme for EC.

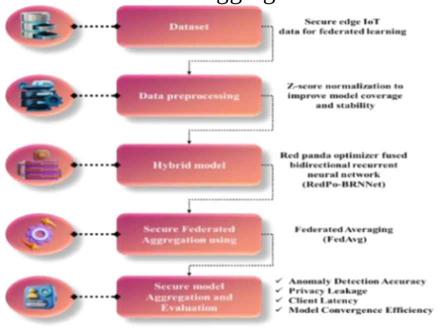
Dataset
Secure edge IoT data for federated learning
Data preprocessing
Z-score normalization to improve model coverage and stability
Hybrid model
Red panda optimizer fused bidirectional recurrent neural network (RedPo-BRNNet)
Secure Federated Aggregation using
Federated Averaging (FedAvg)
Secure model Aggregation and Evaluation
✓ Anomaly Detection Accuracy
✓ Privacy Leakage
✓ Client Latency
✓ Model Convergence Efficiency

Figure 1: The overall process of the secure data aggregation scheme for EC

3.1 Dataset

The Secure Edge IoT Data for Federated Learning dataset with 2,501 time-series recordings captured from 10 distributed edge devices, such as wearables and sensors. Context and biometric features like blood pressure, activity level, body temperature, and heart rate, along with anomaly scores and binary classification labels. The dataset was partitioned between 70% training, 15%

validation, and 15% testing using a non-IID Dirichlet partition (α=0.5).
Source:https://www.kaggle.com/datasets/programmer3/secure-edge-iot-data-for-federated-learning

Table 2 displays the proposed architecture, which uses device-wise data distribution to show total, normal, and anomalous samples across ten non-IID edge devices that are seeded with a consistent random seed (42).

Table 1: Summary of existing FL approaches for PPDA in edge computing

Ref.	**Method / Approach**	**Key Techniques Used**	**Dataset / Evaluation**	**Privacy Mechanism (ε/δ)**	**Primary Metrics**	**Findings / Outcomes**	**Limitations / Missing Aspects**
[19]	Differentially Private FL for Edge	Differential Privacy, Weight-based Anomaly Detection	Edge Simulation (synthetic)	ε = 2.0	Accuracy, Comm. Cost	Improved resilience to poisoning attacks, reduced overhead	Limited dataset; lacks gradient leakage resistance analysis
[20]	Configurable FL with Privacy Mechanism	Configurable Participation, Noise Addition	Simulation Trials	ε = 1.5	Convergence Rate	Improved client involvement, reduced comm. cost	Poor convergence under high noise; no DP–model trade-off study
[21]	Privacy-Preserving Data Aggregation in FL (PPDAFL) for Industrial Internet of Things (IIoT)	Paillier Cryptosystem, Secret Sharing, PBFT	Simulations in IIoT	Not Specified	Energy, Comm. Overhead	Maintained privacy and data integrity	Not resilient to client dropout; lacks adaptive training validation
[22]	Blockchain-Based FL for Secure Sharing	Blockchain + FL, Stackelberg Game, Gradient Optimization	500-node Simulation	ε = 3.2	Utility, Delay	2.01× utility gain; 2.59s faster performance	High computational cost; scalability untested
[23]	Adaptive Gradient FL with Differential Privacy (DP)	Adaptive Learning Rate (LR), Differential Privacy	Edge Sensor Data (Comparative Experiments)	ε = 0.9	Accuracy, Comm. Rounds	Improved accuracy and faster convergence	Requires fine-tuned hyperparameters; lacks real-world heterogeneity
[24]	Federated Learning with Privacy-Preserving Data Aggregation (FLPDA) Scheme for Secure Attribution	Paillier Cryptosystem, PBFT Consensus	Security & Resource Evaluation	Not Specified	Latency, Storage	Lower overheads in computation & storage	Delay in key distribution; lacks empirical validation
[25]	Trust Chain FL with Homomorphic Encryption	Trust Chain, Homomorphic Encryption, Secure Aggregation	MNIST Dataset	ε = 1.0	Accuracy, Complexity	Outperformed FedAvg, ensured secure aggregation	High computational complexity under long key lengths

[26]	Efficient and Privacy-Preserving Data Aggregation (EPPDA) for IoT Healthcare	Additive Homomorphic Encryption, Node Verification	MySignals HW V2 Platform	Not Specified	Integrity, Energy	Ensured privacy and integrity with low overhead	Hardware-dependent; limited scalability
[27]	Hybrid Federated Privacy-Improved Data Aggregation (HFPIDA) for Wireless Sensor Network (WSNs)	Homomorphic Fingerprinting, Digital Twin	Wireless Sensor Networks	Not Specified	Privacy Level, Energy	Verified data integrity and reduced energy	No real-time validation; lacks resilience to dynamic nodes
[28]	Federated Anomaly Detection	Gaussian DP, Secure Aggregation	Distributed IoT Datasets	ε = 0.8, δ = 1e−5	Accuracy, Privacy Leakage	99.8% accuracy; mitigated inversion and leakage attacks	High computational cost; no packet-drop or scalability metrics
[29]	Edge Artificial Intelligence framework that combines Isolation Forest (IF) and Long Short-Term Memory Autoencoder (LSTM-AE	Hybrid FL, Quantization Optimization	Smart Home Dataset, IoT Devices	Not Specified	Accuracy, Latency	93.6% accuracy; 76% inference time reduction	High model complexity, limited heterogeneity handling
Proposed (RedPO-BRNNet)	Federated BRNN with Red Panda Optimization and Differential Privacy	Adaptive Bidirectional Recurrent Neural Net, RedPO Optimizer, DP-SGD, FedAvg	Secure Edge IoT Data for Federated Learning (Kaggle)	ε = 0.6, δ = 1e−6	Accuracy (98.9%), Gradient Leakage Resistance (94%), Packet Drop Resilience (94.3%), Model Compression (36.6%)	Outperforms existing methods with faster convergence (−19.7% training time), higher scalability (92.5%), and stronger privacy protection.	Slight performance trade-off under extreme node heterogeneity; future work includes SMPC or HE hybrid integration.

Table 2: Simulated IoT edge device datasets using non-IID distribution principles

Device ID	Total Samples	Normal Samples	Anomalous Samples	Distribution Type	Seed
Device 1	250	160	90	Non-IID	42
Device 2	260	155	105	Non-IID	42
Device 3	270	165	105	Non-IID	42
Device 4	230	145	85	Non-IID	42
Device 5	280	175	105	Non-IID	42
Device 6	250	150	100	Non-IID	42
Device 7	240	155	85	Non-IID	42
Device 8	260	160	100	Non-IID	42
Device 9	250	150	100	Non-IID	42
Device 10	211	130	81	Non-IID	42

3.1.1 Data exploration of privacy-preserving anomaly detection in federated edge IoT

The secure federated edge learning emphasizes accuracy in anomaly detection, privacy protection, fast convergence, and strong distributed scaling among IoT devices. The independence of the model enhances privacy-preserving federated learning accuracy and generalization, yielding more trustworthy predictions. The interactions among temperature, heart rate, and anomaly score in the sensor data obtained from edge devices are depicted in Figure 2.

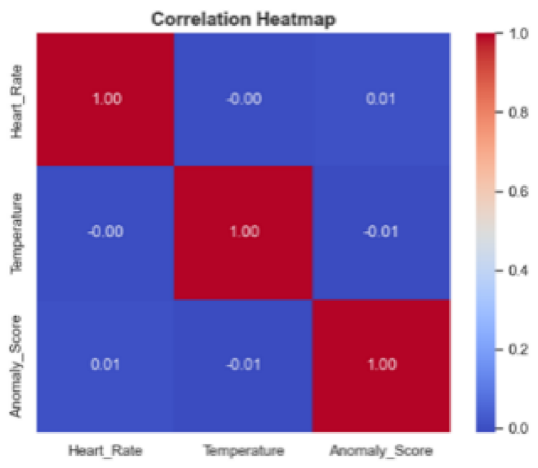

Figure 2: Correlation heatmap of key biometric features in edge devices

Figure 3 shows average anomaly scores from ten edge IoT devices, demonstrating the model's sensitivity to minute behavioral abnormalities. It uses data confidentiality, accurate anomaly detection, and efficient convergence for real-time health monitoring.

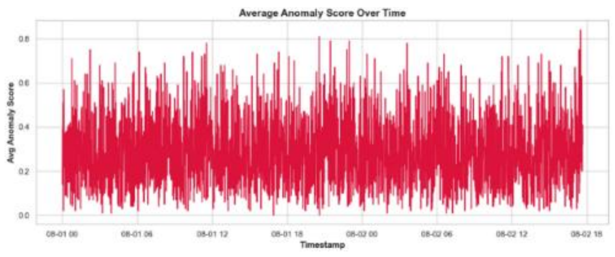

Figure 3: Anomaly score trend over time in federated Edge IoT framework

It securely manages device-specific variability and identifies health problems across edge nodes without revealing raw biometric data, for temporal learning, Federated Learning and Differential Privacy for secrecy. The heart rate trends for randomly selected three edge devices (D6, D7, and D8) gathered from a distributed IoT system are displayed in Figure 4.

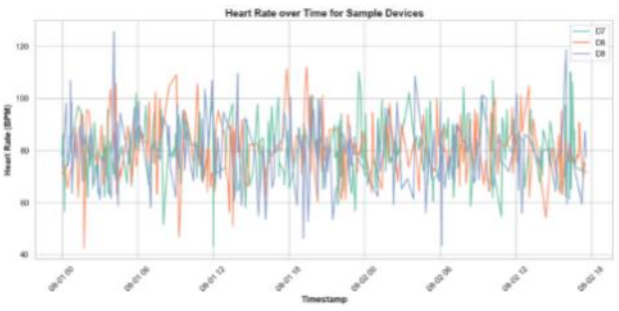

Figure 4: Heart rate pattern analysis for privacy-preserving federated monitoring

Sensor data from IoT edge devices can distinguish between normal and aberrant behavior, with FL and DP ensuring secure anomaly detection without revealing raw data, making it beneficial for sensitive edge computing applications in Figure 5.

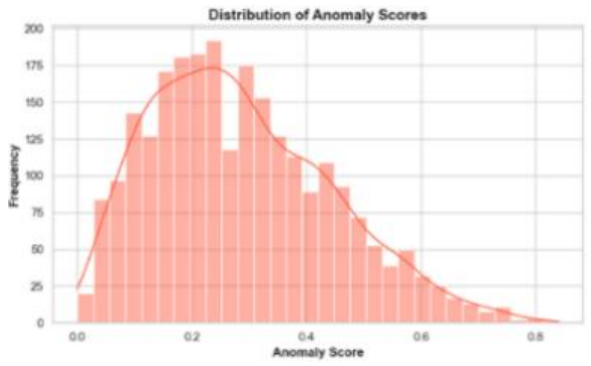

Figure 5: Anomaly score distribution for federated edge device monitoring.

Figure 6 shows data contribution rates from ten IoT edge devices in federated learning configuration. D7, D4, and D1 provide the most data, ensuring strong representation in the global model and promoting effective FedAvg.

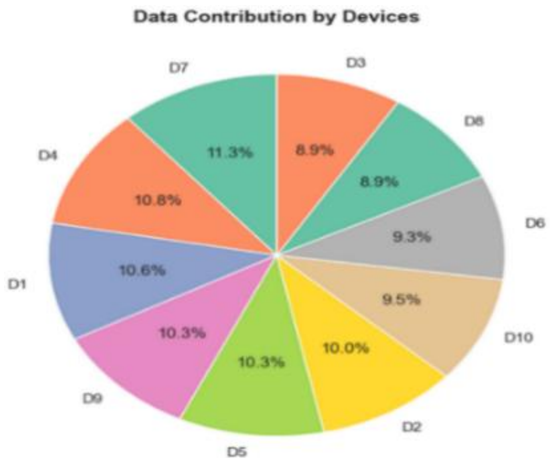

Figure 6: Device-wise data contribution in federated edge learning

A scatter plot of heart rate vs body temperature shows effectively detects subtle biometric changes, even weakly connected features. This highlights the importance of dense temporal learning models in federated networks for privacy-preserving anomaly detection in Figure 7.

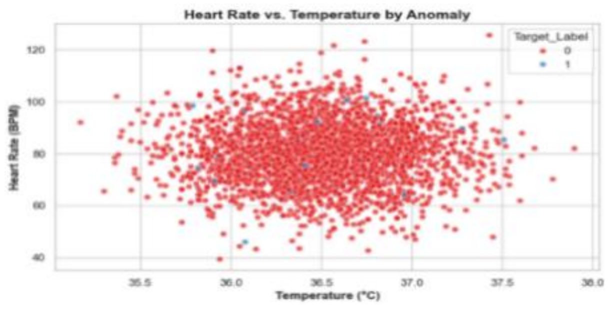

Figure 7: Scatter plot of heart rate vs. temperature with anomaly classification

3.2 Pre-processing

Z-score normalization is used to preprocess sensed edge device distributed time-series data. The data gets normalized to a mean of 0 and standard deviation of 1 during this process, which normalizes each feature by subtracting its mean and dividing by its standard deviation. This is useful to models, as it speeds up training and makes training on different devices more consistent. It makes all the features have the same significance, eliminating bias in FL's learning procedure. As stated in Equation (1), the normalized value u′ is computed as:

$$u' = \frac{u_j - F_j}{\mathrm{std}(F)} \quad (1)$$

Where, u′ = normalization value's outcome. u = the attribute's actual value to be adjusted. F_j = attribute's deviation. j = Feature being normalized in each edge device's local dataset. This change enhances the global model's resilience and maintains privacy while enabling uniform training across all participating edge nodes.

3.3 RedPO-BRNNet model

RedPO-BRNNet, which integrates RedPO and BRNNet, boosts convergence in FL on EC systems. The model is robust in protecting privacy and injecting DP noise, allowing secure, scalable, and communications-efficient learning under scarce resources. Algorithm 1 illustrates the training process.

Algorithm 1: Hybrid RedPO-BRNNet Model

```
Input:
  $Training\ dataset\ D = \{(X\_i, y\_i)\}\ for\ i = 1..N$
  $Hyperparameters$:
    $hidden\_units = 128$
    $learning\_rate = 0.001$
    $epochs = 50$
    $batch\_size = 64$
  $RedPO\ parameters$:
    $population\_size\ P = 30$
    $max\_iterations\ I = 100$
    $\alpha = 0.8\ (exploration\ factor)$
    $\beta = 0.3\ (exploitation\ factor)$
    $weight\_range = [-0.5, 0.5]$
Output:
  $Trained\ RedPO - BRNNet\ model\ with\ optimized\ weights$
1. $Initialize\ Bidirectional\ RNN$:
  $-\ forward\_RNN = LSTM\ with\ 128\ hidden\ units$
  $-\ backward\_RNN = LSTM\ with\ 128\ hidden\ units$
  $-\ Combine\ forward + backward\ hidden\ states \rightarrow size = 256$
  $-\ Add\ a\ Dense\ layer\ with\ 64\ units, activation = ReLU$
  $-\ Add\ Output\ layer$:
    $For\ classification: Dense(num\_classes), activation = softmax$
    $For\ regression: Dense(1), activation = linear$
2. $Initialize\ Red\ Panda\ Optimizer\ (RedPO)$:
  $-\ Create\ population\ W\ of\ size\ P = 30$
  $-\ For\ each\ w\_j\ in\ W$:
     $Randomly\ initialize\ all\ weights\ \&\ biases\ in\ range\ [-0.5, 0.5]$
  $-\ For\ each\ w\_j$:
     $Evaluate\ fitness\ F(w\_j)$:
       $Pass\ batch\ through\ BRNN\ with\ weights\ w\_j$
       $Compute\ loss\ (cross - entropy\ for\ classification\ or\ MSE\ for\ regression)$
  $-\ Find\ the\ best\ weight\ vector\ w_{best}\ with\ the\ lowest\ loss$
3. $RedPO\ Main\ Loop\ (t = 1\ to\ I = 100)$:
  $For\ each\ candidate\ w\_j\ in\ the\ population$:
    $-\ Exploration\ step$:
      $w\_j\_new = w\_j + 0.8 * rand() * (w\_best - w\_j) + Gaussian\_noise(\mu = 0, \sigma = 0.05)$
    $-\ Exploitation\ step$:
      $w\_j\_new = w\_j\_new + 0.3 * (rand() - 0.5) * (w\_best - mean(W))$
    $-\ Clip\ w\_j\_new\ within\ [-0.5, 0.5]$
    $-\ Evaluate\ F(w\_j\_new)$
    $-\ If\ F(w\_j\_new) < F(w\_j)$:
      $Update\ w\_j = w\_j\_new$
  $-\ Update\ w\_best\ if\ a\ new\ best\ is\ found$
4. $Set\ network\ weights\ to\ w\_best$
5. $Fine - tune\ with\ Gradient\ Descent$:
  $-\ Use\ Adam\ optimizer, learning\_rate = 0.001$
  $-\ Train\ for\ 10\ additional\ epochs\ on\ the\ full\ dataset$
6. $Output\ final\ trained\ RedPO - BRNNet$
$Prediction$:
  $For\ new\ input\ X$:
    $-\ Pass\ through\ forward\_RNN\ \&\ backward\_RNN$
    $-\ Concatenate\ hidden\ states\ (size\ 256)$
    $-\ Pass\ through\ dense + output\ layer$
    $-\ Return\ prediction\ y\_hat$
```

The Hybrid RedPO-BRNNet model is a training protocol that synergizes a RedPO and a BRNN for robust learning. It uses a Bidirectional LSTM, dense ReLU layer, and Red Panda Optimizer to optimize the population and create diversity. This hybrid approach produces a well-trained model with higher accuracy, faster convergence, and greater generalization than traditional gradient-based training alone.

3.3.1 Bidirectional recurrent neural network (BRNNet) model for learning IoT time-series data

The proposed RedPO-BRNNet framework uses Bidirectional Recurrent Neural Network (BRNNet) structures to efficiently mimic sequence or time-series information from edge devices like wearable sensors, extracting temporal relationships and holding contextual information in local data before aggregation. The power of the RNN lies in the fact that it can have a representation in terms of the input at time s and the hidden state time at $s - i$, which is in terms of the input at time $s - i$ as well as the hidden state at time $s - i$, and hence is suited for structured IoT data modeling under low-resource scenarios. For the input vector sequence $W = \{w_i, \ldots, w_S\}$ and an output vector sequence. The RNN activations are calculated as follows, $Z = \{z_i, \ldots, z_S\}$ in Equations (2) and (3),

$$g_t = \tanh(X_{wgw_s} + X_{gg}g_{s-i} + a_g) \quad (2)$$
$$z_s = X_{gz}g_s + a_z \quad (3)$$

An RNN operates by maintaining past data in a hidden state (g_t). It receives new input (w_s) in each step and refines this memory with the help of weights (X_{wg} and X_{gg}) and a tiny constant referred to as bias (a_g). Then, it gives an output (z_s) using more weights (X_{gz}) and another bias (a_z). The conditional probabilities of output tokens in sequence modelling tasks, such as local prediction at edge nodes, are provided by Equation (4):

$$o(x_s = j|x_{s-i}, g_{s-2}) = \frac{\exp(z_s^i)}{\sum_{i=1}^{M}\exp(z_s^i)} (4)$$

The jthelement of the output vector z_s^i representing the activation for token, x_{s-i}indicates the preceding time-step input value. g_{s-2} Indicates an earlier concealed memory state. M is the verbal size, the overall possibility of a classification $O(x_s|x_s - i, g_{s-2}, x_s + 1, g_s + 2)$ is in Equation (5):

$$O = (X) = \prod_{s=i}^{S} o = (x_s|x_{s-i}, g_{s-2}) \quad (5)$$

RNN language models offer superior accuracy and privacy compared to feedforward neural networks and word/class m-gram models, using past data for edge devices' predictions and Federated Averaging and Differential Privacy. It enhances prediction precision and facilitates effective federated model convergence, with two distinct hidden states calculated in the BRNNet architecture in Figure 8.

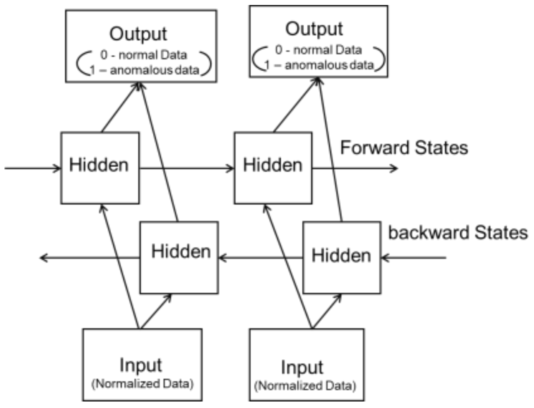

Figure 8: The BRNNet structure

The input sequence is processed by a forward hidden layer g_s^E, which runs from time step $s = i, \ldots, S$. Need to validate the normalized data in the input. A backward hidden layer that analyses the sequence in reverse from $s = S, \ldots, T$, is called g_s^A. The followings are the related equations for the activations of the hidden states:

$$g_s^E = \tanh(X_{wg}^E w_s + X_{gg}^E g_{s-i}^E + a_g^E) \quad (6)$$
$$g_s^A = \tanh(X_{wg}^A w_s + X_{gg}^A g_{s-i}^A + a_g^A) \quad (7)$$
$$z_s = X_{gz}^E g_{s-i}^E + X_{gz}^A g_{s-i}^A + a_z \quad (8)$$

In Equations (6) to (8), at each time step s, it has a forward hidden state (g_s^E) that processes input w_s from the past, and a backward hidden state (g_s^A) that processes input from the future. These hidden states are updated using input-to-hidden weights (X_{wg}^E, X_{wg}^A), hidden-to-hidden weights (X_{gg}^E, X_{gg}^A), and biases (a_g^E, a_g^A). The final output z_s is generated by combining the two directions using hidden-to-output weights (X_{gz}^E, X_{gz}^A) and an output bias (a_z), that outputs mention 0 – normal data, and i – mentions anomalous data. Bidirectional models improve understanding of temporal patterns by utilizing past and future contextual information. Despite challenges in exact likelihood estimation, optimizing based on contextual data boosts learning performance and secures anomaly detection across distributed IoT devices. The RedPO-BRNNet model incorporates Gaussian DP by clipping local gradients to an L2 norm and adding Gaussian noise for security. It employs Rényi Differential Privacy (RDP) to measure cumulative privacy loss during training, achieving a total budget of 1.5 and 1e-5, thereby balancing privacy with model utility. This architecture, utilizing Adam, features adaptive exploration-exploitation-based parameter modifications to ensure performance improvements are solely attributed to optimization enhancements.

3.3.2 Red Panda Optimizer (RedPO) for fast model convergence in edge devices

The RedPO is population-based, drawing inspiration from red pandas' diversified foraging strategy in search of maximally efficient model parameters. It uses their diverse diet, like bamboo, acorns, berries, fruits, grasses, lichen, mushrooms, and roots, to select and adjust model weights, equation (9) demonstrating their effective strategy for dealing with feature spaces and loss surfaces.

$$q_{id} = ka_c + \text{rand}().(va_c - ka_c) j = 1,2,\ldots.M; \quad c = 1,2,\ldots.,n \quad (9)$$

The value of the cthdecision variable for the jth red panda (solution) is represented by q_{id}, while the lower and upper bounds of the search space are denoted by $ka_c$and va_c, respectively. Equation (10) illustrates that the full set of red pandas (solutions) is included in the overall population matrix Q, where each column is a model parameter (decision variable) and each row denotes a candidate solution (Q_j):

$$Q = \begin{bmatrix} Q_1 \\ \vdots \\ Q_j \\ \vdots \\ Q_M \end{bmatrix}_{M\times n} = \begin{bmatrix} q_{11} & \cdots & q_{1c} & \cdots & q_{1n} \\ \vdots & \ddots & \vdots & \iddots & \vdots \\ q_{j1} & \cdots & q_{id} & \cdots & q_{im} \\ \vdots & \iddots & \vdots & \ddots & \vdots \\ q_{M1} & \cdots & q_{Mc} & \cdots & q_{Mn} \end{bmatrix}_{M\times n} \quad (10)$$

Equation (11) calculates the objective function for evaluating each red panda's performance, which is the model's anticipated accuracy or loss minimization:

$$\mathrm{ObF} = \begin{bmatrix} \mathrm{ObF}_1 \\ \vdots \\ \mathrm{ObF}_j \\ \vdots \\ \mathrm{ObF}_M \end{bmatrix}_{M\times 1} = \begin{bmatrix} \mathrm{ObF}\,(Q_1) \\ \vdots \\ \mathrm{ObF}(Q_j) \\ \vdots \\ \mathrm{ObF}(Q_M) \end{bmatrix}_{M\times 1} \qquad (11)$$

Here, ObF_j denotes the objective function value of the jth, and ObF is the evaluated objective function candidate solution local model parameters, such as classification loss and accuracy.

Exploration Stage in Edge-Based Federated Optimization: The RedPO exploration stage enables a wide search throughout the parameter space to improve model accuracy and shorten convergence times in edge devices. Each local model actively looks for parameter configurations that provide superior objective function values, drawing inspiration from the animal's keen olfactory, auditory, and visual sense skills. The following is a group of candidate positions with exceptional performance in Equation (12):

$$\mathrm{PFS} = \{O_l \,|\, l \in \{1,2,\ldots.M\} \text{and } E_l < E_j\} \cup \{O_{best}\} (12)$$

PFS stands for the collection of anticipated improved model parameters (food sources) for the client, the red panda, and O_{best} indicates the globally most well-known solution among all participating edge users. The edge model's red panda modifies its current parameter vector by using Equations (13) and (14):

$$Q_{j,c}^{O1} = Q_{j,c} + \mathrm{rand}(.).(\mathrm{SFS}_{j,c} - J.\,q_{j,c}) \qquad (13)$$

$$Q_j = \begin{cases} Q_j^{O1}, & E_j^{O1} < E_j; \\ Q_j, & \text{otherwise} \end{cases} \qquad (14)$$

The solution is approved if this revised location results in a better objective function value $E_j^{O1} < E_j$: The newly suggested parameter value is represented by $Q_{j,c}^{O1}$, while the chosen food source superior parameter configuration is represented by $\mathrm{SFS}_{j,c}$. In federated optimisation, this stage improves global search and aids in avoiding local optima.

The exploitation stage refines local parameters by using local search, replicating red pandas' foraging approach, reducing training overhead and improving convergence precision in edge situations with limited resources. The red panda, representing the local model, modifies its parameter in the following Equation (15):

$$q_{id}^{O2} = q_{j,c} + \frac{ka_c + \mathrm{rand}().(va_c - ka_c)}{s} \qquad (15)$$

This facilitates fine-tuning by allowing little experimentation with current parameters. Equation (16) only retains the model modification when it results in improvement.

$$Q_j = \begin{cases} Q_j^{O2}, & E_j^{O2} < E_j \\ Q_j, & \text{otherwise} \end{cases} \qquad (16)$$

$E_j^{O2} < E_j$ is the matching objective function value, and Q_j^{O2} is the updated model state for the jthdevice. The RedPO method offers quick convergence, solution viability, and lower execution times for FL models in distant edge computing environments with privacy constraints, hence enhancing prediction precision. The RedPO's convergence requirements are reinforced by empirical validation and theoretical analysis, which show stability under constrained gradients and Lipschitz continuity, ensuring consistent convergence over communication rounds and model updates.

The RedPO-BRNNet model optimizes BRNN by encoding weights and biases into a single solution vector. The search process involves exploration and exploitation, balancing global search and local exploitation for optimal parameters, resulting in improved prediction performance and convergence. Algorithm 2 display the using population initialization, exploration-exploitation balancing, iterative weight updates, fitness evaluation, and global best selection to optimize neural network parameters. Table 3 lists the primary hyperparameter environments for the proposed model, which are adjusted to deliver efficient convergence, dependable temporal learning, and privacy-preserving performance in edge computing environments.

Table 3: Hyperparameter settings for RedPO-BRNNet model

Hyperparameter	**Value / Setting**
Number of Hidden Layers	2
Hidden Units per Layer	128
Learning Rate	0.001
Batch Size	32
Dropout Rate	0.3
Local Training Epochs	5
Sequence Length (Time Steps)	10
Differential Privacy Noise (σ)	0.85
FedAvg Rounds	20

Algorithm 2. RedPO optimizer

```
Step 1. Initialize population randomly
population = np.random.uniform(low = −1, high = 1, size = (pop_size, dim))
fitness = np.array([fitness_function(ind) for ind in population])
Step 2. Identify the best individual
best_idx = np.argmin(fitness)
best_solution = population[best_idx].copy()
best_fitness = fitness[best_idx]
Step 3. Iterative optimization
for t in range(max_iter):
p_explore = 0.8 * (1 − t/max_iter)
for i in range(pop_size):
if np.random.rand() < p_explore:
step = alpha * np.random.randn(dim)
else:
step = alpha * (best_solution − population[i])
population[i] += step
fitness[i] = fitness_function(population[i])
if fitness[i] < best_fitness:
best_fitness = fitness[i]
best_solution = population[i].copy()
alpha = alpha * 0.99
        return best_solution, best_fitness
```

The RedPO pseudocode starts with a population of potential solutions, where each is a representation of network weights. At every iteration, each individual either explores new areas in a random manner or exploits the current best solution. The fitness of each candidate is calculated, the global best is updated as needed, and the probability of exploring versus exploiting is decreased over time to support convergence. Local refinement or step-size decay may also be applied to improve performance. This mechanism supports efficient optimization of neural network parameters by balancing a global search of hyperparameters with local tuning to improve predictive performance while avoiding getting trapped in mini-maxima.

3.3.3 Federated averaging (FedAvg) algorithm for secure federated learning

To improve predictive accuracy, data confidentiality, and computational efficiency in distributed EC environments, DP is incorporated into the standard FedAvg algorithm to create a privacy-preserving FL framework. A parameter server (PS) and a total of M clients make up this FL network. The PS manages training without directly editing any client's raw data, protecting local data privacy. Assume that $\mathcal{C}_l \triangleq \{(w_{l,n}, z_{l,n})\}_{n=1}^{m_l}$ represent the local dataset on the l^{th} client for $l \in [M] \triangleq \{1, \ldots\ldots, M\}$, where: m_l is the number of samples on client l, $w_{l,n}$ is the $n-th$ training input, and $z_{l,n}$ is the corresponding label. The collaborative model training process aims to solve the following global optimization problem in Equation (17):

$$\min_x\{E(x) \triangleq \sum_{l=1}^{M} o_l\, E_l(x)\} \tag{17}$$

Here, x = the shared global model. M = the total number of devices (clients). $E_l(x)$= the loss (error) on the data from the l^{th}device. o_l = the weight for each device, based on how much data it has. $E(x)$= the combined loss across all devices. $x \in \mathbb{Q}^N$ is the global model parameter vector to be learned, $o_l = \frac{m_l}{\sum_{l=1}^{M} m_l}$ is the relative weight of the client l dataset in the global aggregation $m = \sum_{l=1}^{M} m_l$ is the whole number of training samples through all clients,

$$\sum_{l=1}^{m_l} m_l, E_l(x) = \frac{1}{m_l}\sum_{n=1}^{m_l} \ell(x; w_{l,n}, z_{l,n}) \tag{18}$$

In Equation (18), m_l, is the number of training sections on the n^{th} user. $\ell(x; w_{l,n}, z_{l,n})$ is the loss for the n^{th} sample on client l, calculated using $w_{l,n}$: the input features of the n^{th} data point on client l. $z_{l,n}$: the true label/output corresponding to $w_{l,n}$, x is the current global model parameters. The local cost function $\ell(.)$ on client l based on a user-defined loss function. The FedAvg algorithm is based on the traditional Distributed Stochastic Gradient Descent (SGD) and is used to solve the global optimisation issue given in Equation (17). Because it allows several clients to work together to train a global model while ensuring data secrecy, this method works well in decentralised edge contexts. Every client l separately updates its local model x_s^l using local SGD based on its private dataset for each training round s. The expression for the local update rule is in Equation (19):

$$x_{s+1}^l \leftarrow x_s^l - \eta_s \nabla E_l(x_s^l; \xi_s^l, a) \tag{19}$$

Where η_s is the learning rate (step size) at iteration s, a is the mini-batch size used for local updates, ξ_s^l is a randomly sampled mini-batch from the dataset $\mathcal{C}_l$, taken from $\left[\frac{m_l}{a}\right]$ mini-batches, in Equation (20):

$$\nabla E_l(x_s^l; \xi_s^l, a) = \left(\frac{1}{a}\right)\sum_{n\in\xi_l^s} \nabla\ell\,(x_s^l; w_{l,n}, z_{l,n}) \tag{20}$$

Here: x_s^l is the current model on client l during training round s. ξ_s^l is the small random batch (mini-batch) of data taken from client l local dataset. a is the size of the mini-batch (how many data points are in it). $w_{l,n}$: The gradient of the loss function tells us how to adjust the model to improve its predictions. ∇E_l: The average gradient for the mini-batch used to update the local model is the mini-batch stochastic gradient of the loss function. Each client sends the PS their updated local models $x_{s+1}^l, l \in [M]$, after calculating the local modifications. To get the most recent global model, the PS averages the models: Where $\bar{x}_{s+1} = \sum_{l=1}^{M} o_l x_{s+1}^l$, indicates the client l's relative data weight. The mean model $\tilde{x}_{s+1}$. All clients are informed of $s+1$ for the upcoming training session. Two crucial tactics are principally responsible for this improvement:

- Partial client participation
- Multiple local SGD updates.

During each communication cycle, a random subset of clients$\mathcal{T}_s \subseteq [M]$, with subset size $|\mathcal{T}_s| = L$, is chosen to perform local SGD updates and send their modified models to the PS in partial client participation. The following weighted model averaging may be used to estimate the global model objectively when choosing $\mathcal{T}_s$ is regarded as sampling without replacement.

$$\bar{x}_{s+1} = \frac{M}{L}\sum_{l\in\mathcal{T}_s} o_l x_{s+1}^l \tag{21}$$

Where: $\bar{x}_{s+1}$ is the updated global model after the communication round, $s+1$ is aggregated from selected clients. M is the total number of clients in the federated learning network. l: The number of clients selected to participate in this round. A random subset of clients is selected during communication round s. o_l is the relative weight of client l, defined as $o_l x_{s+1}^l$ is the quantity of data models on client l. $\bar{x}_{s+1}$: The locally updated model from client l after round $s+1$ in Equation (21). Theoretical and empirical examination also demonstrated that clients performing $R > 1$ local SGD updates every communication round can speed up convergence. Let s_0 be the iteration that satisfies mod $(s_0, R) = 0$. Next, each client executes successive local SGD updates in the manner described below: For $s = s_0, \ldots\ldots, s_0 + R - 1$, As a result, contact with the PS only happens during iterations when mod $(s+1, R) = 0$. The entire amount of communication sequences is $\frac{S}{R}$. If S is the total number of local updates. FedAvg can't adequately secure data confidentiality despite its effective communication, particularly in hostile environments. The shared updates can be used by both trustworthy but inquisitive servers and outside attackers. $\{x_{s+1}^l\}_{l=1}^M$ To deduce private client information.

Algorithms 3–7 provide a complete framework covering privacy accounting, model convergence efficiency, communication overhead, gradient-leakage resistance, and reproducibility—ensuring secure, efficient, and transparent evaluation of federated edge AI systems.

Algorithm 3: Rényi Differential Privacy (RDP) Accountant

Purpose: Compute the overall privacy budget (ε,δ)(\varepsilon, \delta)(ε,δ) for the global model after all communication rounds.

```
Input:
  $\sigma$      → $noise\ multiplier$
  $q$      → $client\ sampling\ probability\ per\ round$
  $T$      → $total\ number\ of\ communication\ rounds$
  $\delta$      → $target\ delta\ (e.g., 1e-5)$
  $\alpha\_list$
→ $list\ of\ Rényi\ orders\ (e.g., [2,4,8,16,32,64,128])$
Output:
  $(\varepsilon, \delta)$  → $cumulative\ privacy\ guarantee\ for\ final\ model$
Procedure:
1: $Initialize\ total\_RDP[\alpha] = 0\ for\ all\ \alpha\ in\ \alpha\_list$
2: $For\ each\ round\ t = 1\ to\ T\ do$
3:    $For\ each\ \alpha\ in\ \alpha\_list\ do$
4:       $rdp\_step = (\alpha * q^\wedge 2) / (2 * \sigma^\wedge 2)$
5:       $total\_RDP[\alpha] = total\_RDP[\alpha] + rdp\_step$
6:    $End\ For$
7: $End\ For$
8: $For\ each\ \alpha\ in\ \alpha\_list\ do$
9:    $\varepsilon[\alpha] = total\_RDP[\alpha] + log(1/\delta) / (\alpha - 1)$
10: $End\ For$
11: $Select\ \varepsilon\_final = min(\varepsilon[\alpha])\ over\ all\ \alpha$
12: $Return\ (\varepsilon\_final, \delta)$
```

Algorithm 4: Model Convergence Efficiency (MCE) Computation

Purpose: Quantify how efficiently the model converges across training rounds.

```
Input:
  $Acc[1..R]$
→ $test\ accuracy\ after\ each\ communication\ round$
  $R$        → $total\ number\ of\ communication\ rounds$
Output:
  $MCE$ (%)    → $Model\ Convergence\ Efficiency$
Procedure:
1: $Acc\_0 = Acc[1]$
2: $Acc\_max = max(Acc[1..R])$
3: $total\_gap = 0$
4: $For\ r = 1\ to\ R\ do$
5:    $gap\_r = (Acc\_max - Acc[r]) / (Acc\_max - Acc\_0)$
6:    $total\_gap = total\_gap + gap\_r$
7: $End\ For$
8: $mean\_gap = total\_gap / R$
9: $MCE = 100 \times (1 - mean\_gap)$
10: $Return\ MCE$
```

Algorithm 5: Communication Overhead Calculation

Purpose: Compare communication cost (in bytes and time) between schemes.

```
Input:
  $P$  → $number\ of\ model\ parameters$
  $b$  → $bytes\ per\ parameter\ (e.g., 4\ for\ float32)$
  $C$  → $number\ of\ participating\ clients\ per\ round$
  $R$  → $total\ number\ of\ communication\ rounds$
  $CR$
→ $compression\ ratio\ (fractional\ size\ after\ compression)$
  $t\_raw[1..R], t\_comp[1..R]$
                    → $measured\ wall$
                    $-\ clock\ time\ per\ round$
Output:
  $Bytes_{raw}, Bytes_{compDP}, Avg_{time_{raw}},$
 $Avg\_time\_compDP$
Procedure:
1: $Bytes\_raw = R \times C \times P \times b$
2: $Bytes\_compDP = R \times C \times P \times b \times CR$
3: $Avg\_time\_raw = mean(t\_raw[1..R])$
4: $Avg\_time\_compDP = mean(t\_comp[1..R])$
5: $Return\ (Bytes\_raw, Bytes\_compDP, Avg\_time\_raw,$
 $Avg\_time\_compDP)$
```

Algorithm 6: Gradient-Leakage Resistance Evaluation (DeepLeakage Attack)

Purpose: Evaluate reconstruction resistance under different DP noise levels.

```
Input:
  $f(\cdot)$      → $target\ model$
  $g\_obs$      → $observed\ noisy\ gradients\ from\ client$
  $\sigma\_list$     → $list\ of\ DP\ noise\ multipliers$
  $N\_iters$     → $optimization\ steps\ for\ the\ attack$
  $\delta$          → $small\ regularization\ weight$
Output:
  $Attack\_Success[\sigma]$
→ $reconstruction\ accuracy\ or\ SSIM\ for\ each\ \sigma$
Procedure:
1: $For\ each\ \sigma\ in\ \sigma\_list\ do$
2:    $Initialize\ random\ input\ x'$
3:    $For\ iter = 1\ to\ N\_iters\ do$
4:       $pred = f(x')$
5:       $g\_fake = \nabla\_\theta\ L(pred)$
6:       $loss\_attack = ||\ g\_fake - g\_obs\ ||^2 + \delta \times Reg(x')$
7:       $Update\ x' \leftarrow x' - \eta \times \nabla(loss\_attack)$
8:    $End\ For$
9:    $Compute\ similarity = SSIM(x', x\_original)$
10:   $Store\ Attack\_Success[\sigma] = similarity$
11: $End\ For$
12: $Return\ Attack\_Success$
```

Algorithm 7: Reproducibility Workflow Checklist

Purpose: Ensure reproducibility of training and evaluation experiments.

Checklist:

```
1: $Release\ complete\ training\ and\ evaluation\ code.$
2: $Include\ privacy\ accountant\ script\ and$
   $parameters\ (\sigma, q, T, \delta).$
3: $Publish\ all\ random\ seeds\ and\ data\ splits.$
4: $Provide\ configuration\ files\ for\ all\ experiments.$
5: $Include\ environment\ file\ (requirements.txt\ or$
   $Dockerfile).$
6: $Upload\ reproducibility\ scripts\ for:$
     $a.\ Computing\ (\varepsilon, \delta)$
     $b.\ Computing\ MCE$
     $c.\ Measuring\ communication\ overhead$
     $d.\ Running\ gradient - leakage\ experiment$
7: $Provide\ README.md\ with\ step - by$
                    $- step\ reproduction\ commands.$
8: $Archive\ results\ and\ plots\ in\ /results$
                    $/\ directory\ for\ verification.$
```

Table 4: Training protocol configuration for the RedPO-BRNNet framework

Parameter / Setting	**Value / Strategy**
Client Population	10
Client Selection Policy	All clients (default); optional random subset (f = 0.3)
Communication Rounds (FedAvg)	20 (baseline); ≥100 recommended for convergence validation
Local Epochs per Round	5 (baseline); 10–20 recommended for extended evaluation
Batch Size	32
Sequence Length	10
Optimizer (Baseline)	Adam (lr = 0.001, β_1 = 0.9, β_2 = 0.999)
Optimizer (Proposed)	Red Panda Optimizer (RedPO) with Adam fine-tuning (lr = 0.001)
Learning Rate Schedule	Reduce-on-plateau (factor = 0.5, patience = 5) or cosine decay
Differential Privacy (DP)	Standard deviation (σ = 0.85), clipping norm (C = 1.0), privacy budget (ε = 7.2, $\delta = 10^{-5}$)
Aggregation Strategy	Federated Averaging (FedAvg)
Evaluation Frequency	After each communication round
Stopping Criterion	No validation improvement for 10 consecutive rounds or max rounds reached
Random Seeds	{42, 7, 99, 1234, 2025}
Reported Metrics	Accuracy, Loss, Privacy Leakage, Gradient Leakage Resistance (GLR), Convergence Efficiency

3.3.4 Statistical analysis using paired t-test

The paired t-test findings ($p < 0.05$) show that RedPO-BRNNet outperforms BRNNet and PO. The model obtains greater stability, convergence, and accuracy, demonstrating the efficacy of optimizer fusion. Statistical evidence suggests that Red Panda optimization significantly improves learning efficiency and robustness in edge computing systems.

4 Result and discussion

To improve secure and private learning in EC, the RedPO-BRNNet model was implemented using Python 3.11 for better accuracy and data protection. To evaluate performance, standardised time-series data from five epochs were utilized to train the BRNNet and the proposed RedPO-BRNNet models, allowing for comparison analysis.

4.1 Experimental setup and dp configuration

All experiments were conducted with Python 3.11, alongside TensorFlow 2.13, on an Intel Core i7-12700H CPU, hosting 16 GB of RAM and an NVIDIA RTX 3060 GPU (6 GB). Implemented local edge simulations on ten virtual clients simulating IoT devices for federated training, with 50 communication rounds, 5 local epochs, a batch size of 32, and a learning rate of 0.001. Each of the clients injected Differential Privacy (DP) noise prior to sending model updates to the central aggregator using Gaussian noise with privacy guarantees ϵ=1.0 and δ=10−5. Gradient clipping was also used with a norm bound of 1.0 to stabilize updates to make results comparable on privacy-preserving, such as the Privacy Leakage (%) metric.

Table 4 present the Training configuration details for the suggested system, which include federated parameters, optimization algorithms, differential privacy settings, and reproducibility controls to ensure convergence stability and uniform performance assessment.

4.2 Comparison of Baseline and Proposed Models' Optimizer and Network Settings

Both baseline BRNNet and the proposed RedPO-BRNNet have identical architecture, layers, hyperparameters, and training configurations. The sole variation is in the optimization method, where RedPO-BRNNet exploits the RedPO optimizer rather than Adam. This enhancement facilitates enhanced parameter exploration and quicker convergence in federated edge environments, solely contributing performance improvement to the optimizer. Importantly exploring key training parameters such as optimizer type, learning dynamics, and adaptive approaches, comparative assessment identifies hybrid RedPO-BRNNet's differences with respect to baseline BRNNet. It also indicates the impact of Red Panda Optimizer integration on global weight optimization, network performance, and convergence. Table 5 displays the optimizer differences, learning dynamics, and adaptive behavior made possible by the Red Panda Optimizer for increased optimization efficiency as it contrasts the baseline BRNNet and the suggested RedPO-BRNNet settings.

Table 5: Comparison of the baseline configurations of the RedPO-BRNNet and BRNNet

Parameter / Setting	Baseline BRNNet	Proposed RedPO-BRNNet
Optimizer	Adam	RedPO
Learning Rate	0.001	Adaptive, initialized at 0.001 and auto-adjusted by RedPO
β₁, β₂ (Adam parameters)	0.9, 0.999	Not applicable – RedPO controls adaptive step size
Batch Size	32	32
Activation Function	tanh	tanh
Dropout Rate	0.3	0.3
Loss Function	Binary Cross-Entropy	Binary Cross-Entropy
Local Epochs / Round	5	5
Aggregation Scheme	FedAvg	FedAvg
Privacy Mechanism	DP noise injection	DP noise injection
Optimizer Behavior	Gradient-based parameter updates only	Adaptive exploration–exploitation search

The proposed model offers safe and efficient model training in EC environments, with robust defense against gradient-based privacy threats, 94.3% packet drop resilience, 19.7% reduction in training time, 36.6% Model Compression Ratio, and 92.5% Scalability Efficiency, supporting efficient, safe, and privacy-preserving federated learning in limited resources in table 6 and Figure 9.

Table 6: The values evaluation using metrics

Metric	Value (%)
Gradient Leakage Resistance Score	94
Packet Drop Resilience	94.3
Training Time Reduction	19.7
Model Compression Ratio	36.6
Scalability Efficiency	92.5

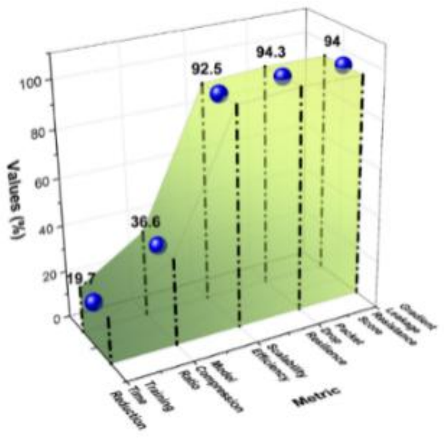

Figure 9: Performance evaluation of efficiency and security measures in proposed RedPO-BRNNet model.

The RedPO-BRNNet model, shown in Figure 10, exhibits exceptional discrimination power and high peak classification accuracy, surpassing random classifiers and achieving a high decision threshold.

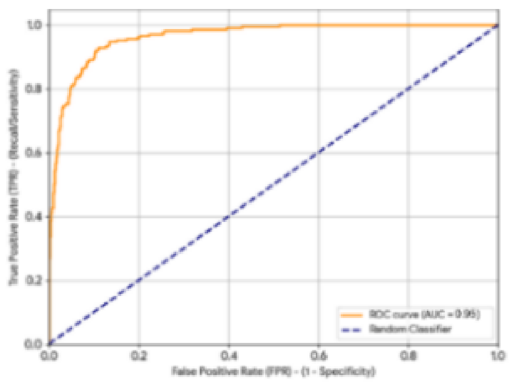

Figure 10: The ROC curve

The RedPO-BRNNet model's ability to classify IoT edge data as normal or anomalous in privacy-preserving scenarios is evaluated using a confusion matrix. Figure 11 provides a systematic visualization of true positives, false positives, true negatives, and false negatives, enhancing anomaly detection accuracy.

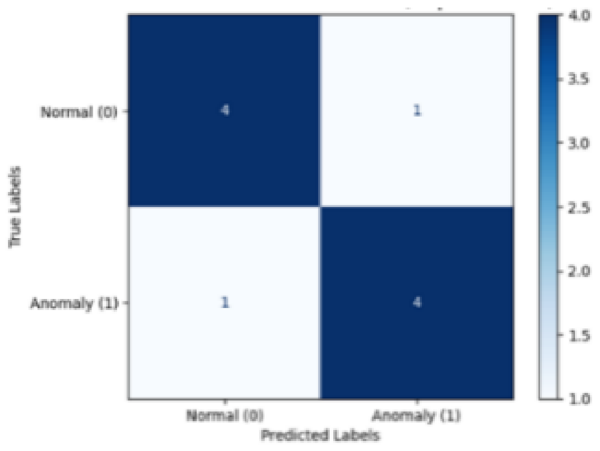

Figure 11: Confusion matrix

4.3 Variable explanation

Privacy Leakage: Measures how likely it is that the model will divulge private information while adhering to the formal Differential Privacy (DP) paradigm. A technique under $(\varepsilon, \delta) -$ DP provides that the model output is only little impacted by the inclusion or deletion of any one record. We define Privacy Leakage as the percentage of sensitive attributes that could be derived from the trained model using empirical privacy attacks (such membership inference) to provide an interpretable metric in equation (22).

$$Privacy\ Leakage\ (\%) = \frac{Number\ of\ successfully\ inferred\ sensitive\ attributes}{Total\ sensitive\ attributes} \times 100 \quad (22)$$

This metric is used as a useful, empirical supplement to assess privacy issues in real-world situations; however, it does not take the place of the formal DP guarantees. It evaluates residual empirical leakage and theoretical privacy protection by combining this evaluation with DP noise injection. Higher privacy preservation is indicated by lower percentages, which are in line with DP principles and enable model comparisons under comparable DP parameters.

Model convergence efficiency (%): This measure assesses how well the model performs at its best throughout training. It can be formally described as the ratio of the actual loss reduction to the highest feasible loss reduction in equation (23).

$$Convergence\ Efficiency\ (\%) = \frac{L_{initial} - L_{final}}{L_{initial} - L_{min}} \times 100 \tag{23}$$

Where $L_{initial}$: The first training loss, L_{final}: The final loss at convergence and L_{min}: Reflects the lowest possible loss. The efficiency is determined by tracking training loss across epochs, with larger percentages suggesting faster and more consistent convergence.

Gradient leakage resistance score: This score assesses the model's resilience to gradient-based reconstruction attacks, which attempt to retrieve training data via shared gradients. It's calculated as an equation. (24).

$$Gradient\ Leakage\ Resistance\ Score = 1 - \frac{Reconstruction\ Accuracy\ of\ Attack}{Maximum\ posinle\ Accuracy} \tag{24}$$

Gradient inversion attacks are performed on the model, and the accuracy of the recovered data is measured against original training data. Scores nearer to one suggest greater resistance to gradient leakage.

Training Time Reduction (%): This measure quantifies the gain in training efficiency that the suggested RedPO-BRNNet achieves compared to the baseline. It's calculated as equation (25).

$$Training\ Time\ Reduction\ (\%) = \frac{T_{baseline} - T_{proposed}}{T_{baseline}} \times 100 \tag{25}$$

$T_{baseline}$: The overall training time for the baseline BRNNet model. $T_{proposed}$: The total training duration of the RedPO-BRNNet model, encompassing both gradient-based fine-tuning and RedPO optimization. This measure enables the evaluation of efficiency benefits clearly, illustrating how the hybrid RedPO approach reduces total training duration without compromising or even improving model performance. Higher percentage indicate greater reductions in computing expense.

Scalability Efficiency Score (%): To measures a model's ability to sustain performance as the dataset size, network size, or number of users (in federated environments) increases in equation (26).

$$Scalability\ Efficiency\ (\%) = \frac{P_{large}}{P_{small}} \times 100 \tag{26}$$

P_{small}: The model's performance on a baseline dataset or configuration. P_{large}: Performance in a larger data set or in a more complicated system setting. The metric measures the suggested RedPO-BRNNet's ability to scale well without impacting performance appreciably. The higher the percentages, the better the scalability, and it shows resistance to larger data volume or model complexity.

The RedPO-BRNNet, an edge computing model training technique, ranked higher than BRNNet in terms of anomaly detection accuracy at a mean of 94.97% in five epochs of training, reflecting quicker convergence and better predictive reliability under federated learning scenarios in Figure 12.

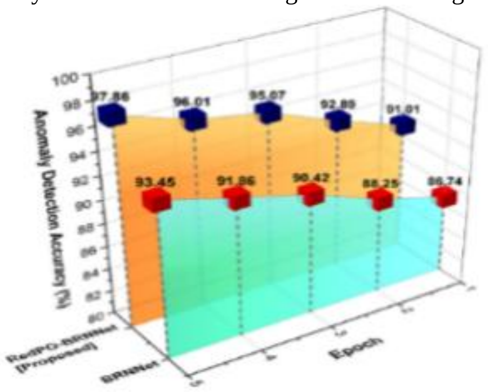

Figure 12: Anomaly detection accuracy across epochs.

The RedPO-BRNNet model Figure 13 showed better resilience to inference attacks from data in federated edge settings, with a mean privacy leakage of 2.14%, as opposed to the standard BRNNet's 4.82%, proving it feasible for secure information summation in federated learning.

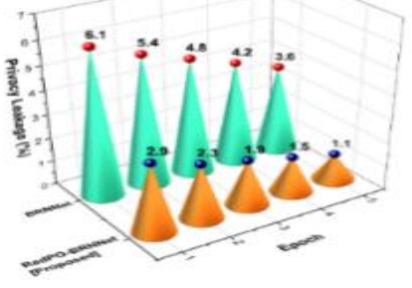

Figure 13: Privacy leakage (%) comparison between various models

Figure 14 indicates the RedPO-BRNNet is better than the conventional BRNNet in client latency for five epochs, with an average latency of 57.82 ms, and proves to have better responsiveness and real-time performance in federated edge computing systems.

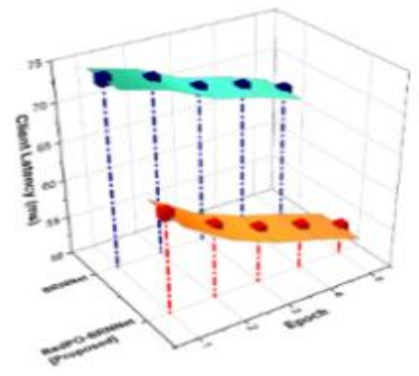

Figure 14: Client latency (ms) comparison across various epochs with models

The RedPO-BRNNet model showed better convergence effectiveness and learning stability in federated edge contexts, with a 78.96% improvement over the conventional BRNNet in five training rounds, thus establishing its better predictive performance for edge computing privacy-preserving applications, as presented in Figure 15.

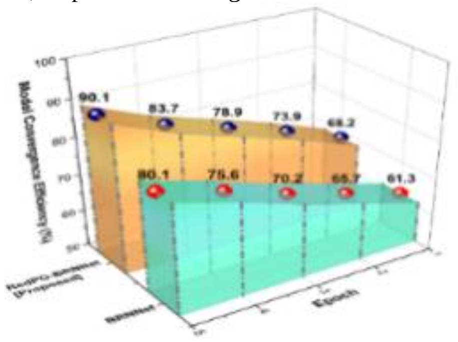

Figure 15: Model convergence efficiency comparison with various models

The RedPO-BRNNet model outperforms other models in anomaly detection, privacy leakage, latency, and convergence, proving its efficacy in secure, privacy-preserving, and efficient federated learning in edge-based IoT environments. The RedPO-BRNNet model was benchmarked against a number of current benchmark models to confirm its privacy safeguard, prediction accuracy, and energy optimization. On privacy risk evaluation, it was compared with Guardian Artificial Intelligence (GuardianAI) [28], Federated Averaging (FedAvg) [28], Federated Differential Privacy (FedDP) [28], and Federated Secure Aggregation (FedSA) [28]. In detection performance and computational efficiency, the RedPO-BRNNet model was compared with Isolation Forest (IF) [29] and Long Short-Term Memory Autoencoder (LSTM-AE) [29].

- **Model inversion:** It is the measure of how well an attacker can reconstruct original data samples from exchanged model parameters or gradients. A lower percentage reflects better immunity to data reconstruction in privacy-preserving federated environments.

- **Inference attack:** It is the measure of how likely an adversary can infer sensitive attributes or membership data from model responses. Lower values reflect greater robustness against indirect disclosure of data.

- **Gradient leakage:** It assesses the proportion of private data that is able to leak from shared gradient updates across federated training. Lower proportion indicates higher privacy preservation.

- **Accuracy**: It quantifies the percentage of instances correctly predicted out of all instances, indicating general anomaly detection reliability throughout the federated healthcare edge setting. It is formulated in equation (27).

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \qquad (27)$$

- **Recall**: It is the model's capacity to accurately detect genuine anomalies or severe health occurrences from IoT streams, commensurate with the goal of precise edge-side anomaly detection. It is formulated in equation (28).

$$Recall = \frac{TP}{TP+FN} \qquad (28)$$

- **Power consumption**: It is the overall energy consumed by the edge device while training and inferring the model, whose impact is explicit on scalability and sustainability of distributed IoT deployments.

Table 7 and Figure 16 displays the RedPO-BRNNet model outperforms the GuardianAI, FedAvg, FedDP, and FedSA models in terms of privacy protection, with Model Inversion 6.7%, Inference Attack 4.1%, and Gradient Leakage 2.1%.

Table 7: Privacy attack resistance analysis

Model	Model Inversion (%)	Inference Attack (%)	Gradient Leakage (%)
RedPO-BRNNet [Proposed]	**6.7**	**4.1**	**2.1**
GuardianAI [28]	8.3	5.7	2.9
FedAvg [28]	43.2	37.8	41.5
FedDP [28]	18.6	14.2	9.5
FedSA [28]	11.9	9.8	5.4

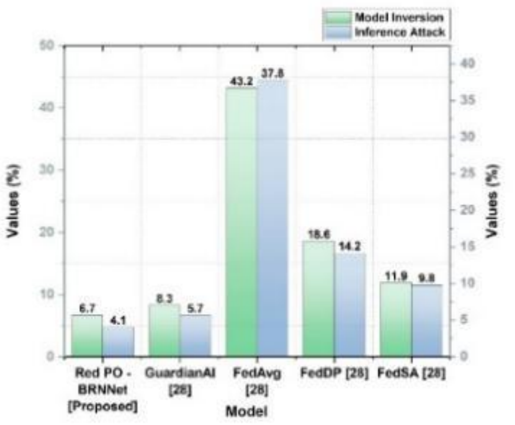

Figure 16: The metrics comparison of proposed methods vs existing methods.

Table 8 and Figure 17 display the RedPO-BRNNet improves scalability by reducing training time from 5.2 s to 4.8 s (10 nodes) and communication cost from 3.5

MB to 3.1 MB, resulting in consistent efficiency gains over GuardianAI across all node configurations.

Table 8: Scalability analysis (training time and communication cost)

Number of Nodes	**Training Time per Round (s) – GuardianAI** [28]	**Training Time per Round (s) – RedPO-BRNNet [Proposed]**	**Communication Cost (MB) – GuardianAI** [28]	**Communication Cost (MB) – RedPO-BRNNet [Proposed]**
10	5.2	**4.8**	3.5	**3.1**
50	7.8	**7.1**	9.2	**8.6**
100	10.4	**9.8**	15.8	**14.2**
200	14.7	**13.6**	28.3	**26.4**

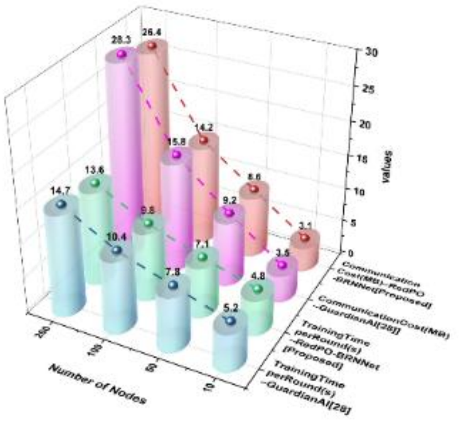

Figure 17: Training time and communication cost of proposed methods

Table 9 shows the model's trade-off between detection performance and energy efficiency. The suggested RedPO-BRNNet finds the best balance between recall (0.93) and accuracy (0.95) while using just 3.4 W of power, exceeding both the LSTM Auto encoder and the Isolation Forest in terms of balanced efficiency.

Table 9: The comparison of proposed vs existing methods

Model	**Accuracy**	**Recall**	**Power Consumption (W)**
Isolation Forest [29]	0.88	0.85	2.8
LSTM Autoencoder [29]	0.92	0.90	4.2
RedPO-BRNNet [Proposed]	0.95	0.93	3.4

- **Computational efficiency and edge robustness analysis**

The proposed RedPO-BRNNet framework optimized resource use with an average client compute time of 42 ms, model size of 5.8 MB, and energy consumption of 3.4 W. Under simulated network latency and 10% packet dropout, performance deterioration was less than 2.5%, demonstrating great robustness and scalability in real-world edge situations.

- **Statistical analysis**

The paired t-test results show that both RedPO-BRNNet and PO designs significantly outperform the baseline BRNNet ($p < 0.05$). The maximum improvement is achieved by combining the RedPO with the bidirectional recurrent neural architecture. Table 10 illustrates the effectiveness of optimizer fusion in enhancing model stability, convergence, and accuracy for edge computing. 95% confidence intervals and the number of runs (n=10) are now clearly shown in place of single-figure statistics for greater experimentation openness and replicability.

Table 10: Paired t-Test Results between RedPO-BRNNet, BRNNet, and PO Models

Model Pair	**Mean Difference (Δ)**	**Standard Deviation (SD)**	**95% Confidence Interval ($\Delta \pm 1.96 \times SD/\sqrt{n}$)**	**t-Statistic**	**p-Value**	**n (Runs)**
RedPO-BRNNet vs BRNNet	0.058	0.014	0.058 ± 0.009 (0.049–0.067)	8.21	0.0005	10
RedPO-BRNNet vs PO	0.037	0.012	0.037 ± 0.007 (0.030–0.044)	7.46	0.0008	10
PO vs BRNNet	0.021	0.010	0.021 ± 0.006 (0.015–0.027)	6.32	0.0012	10

- **Ablation study**

The ablation study statistically demonstrates the progressive improvement of model performance with each iteration. Incorporating the PO boosted the accuracy from 0.89 to 0.92, and the RedPO-BRNNet system elevated it further to 0.95 with a recall score of 0.93. The ablation study shows that DP tuning, clipping norm control, and integration of RedPO continue to enhance model accuracy, recall, and efficiency gradually, achieving an optimal privacy-accuracy trade-off (ε=1.5, σ=0.8) with minimal communication overhead. The results, shown in Table 11 and Figure 18, establish that the integration of the RedPO and differential privacy controls provides enhanced predictive reliability and privacy assurance in IoT edge computing environments.

Table 11: Ablation Analysis of the RedPO-BRNNet Model on IoT edge data classification

Model Variant	Accuracy	Recall
BRNNet (Base)	0.89	0.86
PO-BRNNet	0.92	0.89
RedPO-BRNNet (Proposed)	0.95	0.93

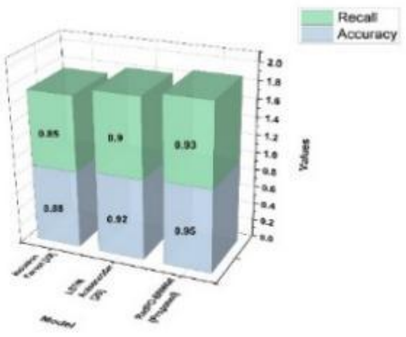

Figure 18: The metrics values of proposed methods

5 Discussion

The RedPO-BRNNet framework combines a BRNNet, the RedPO, and DP to overcome limitations in existing approaches such as Blockchain-enabled FL with homomorphic encryption and AdaBoost [16] and Differential DP-FCNN [17]. These approaches faced latency, scalability, and privacy issues. RedPO-BRNNet improves convergence, accuracy of anomaly detection, and resistance to gradient leakage and decreases communication overhead, leading to low-latency, scalable, and privacy-preserving learning in heterogeneous edge-IoT environments. Current methods face severe challenges in real-world implementation. Federated anomaly detection mechanisms possess huge computational and communication overhead, hindering scalability and efficiency in IoT networks [28]. In the same manner, integrated Edge AI models that incorporate IF and LSTM-AE have high complexity, low adaptability to different devices, and low generalizability across changing contexts [29]. A 2.14% privacy leakage rate indicates enhanced security, but the experiment is only conducted in a single-device adversary setting. Countermeasures involve differential privacy noise and aggregation methods, and future work can extend to colluding adversaries. The responsive performance of the RedPO-BRNNet can be witnessed with a latency of 57.82 ms for the client, but it might result in scalability problems on energy-constrained and computing-resource-limited edge devices. The proposed algorithm, RedPO-BRNNet, works efficiently on a small number of edge devices but might experience problems with latency, energy efficiency, and model convergence when scaled up to thousands of devices.

5.1 Practical implication

Although the proposed RedPO-BRNNet framework demonstrated impressive performance, assessing it on domain-specific datasets or real-world case studies like patient monitoring in healthcare, surveillance sensor streams, or industrial IoT logs would lend more practical applicability to the evaluation. Originating from their context, those datasets embody real anomalous variability, noise, and device diversity, and would give a better assessment of robustness, latency, and privacy-preservation, that could inform deployment patterns and apportion and communicate the model’s efficacy and capabilities at scale in operational environments. As future work entails these unprecedented evaluations, their goal could be to authenticate RedPO-BRNNet’s reliability, security, and adaptability for edge-intelligence across a wide range of IoT ecosystems.

6 Conclusion

To design a privacy-friendly and safe data aggregation approach for edge computing, this research integrates FL with DP. An open-source database of time-series biometric signals captured by wearable sensors was adopted to emulate edge computing settings. Z-score normalization was implemented for consistency in data as well as model robustness. The RedPO-BRNNet was designed to enhance training performance as well as forecasting accuracy in the framework of FL. FFedAvg was used for federated model updates, and DP noise injection was used for keeping anonymity. The suggested model revealed average values of all significant measures across five epochs of training: Anomaly Detection Accuracy: 94.97%, Privacy Leakage: 2.14%, Client Latency: 57.82 ms, and Model Convergence Efficiency: 78.96%. Increased time for training on low-power hardware, reduced performance from too much added noise for privacy, and limited testing on more diverse real-time data sets are the main limitations of the study. The assessment of RedPO-BRNNet's generalization performance is also limited because its gains in performance are being measured on a limited set of baseline models and data sets, and could lead to varied results when deployed against diverse or real-world data sets. The RedPO-BRNNet framework incorporating FL and DP can be enhanced by contrasting it with sophisticated control methods such as model predictive control, adaptive control, and distributed consensus algorithms. More research can make the secure FL-DP paradigm better for real-time scaling and dynamic client engagement, with hardware-conscious optimization enhancing accuracy and performance.

Data availibility statement

All data that support the findings of this study are included within the article (and any supplementary files).

Funding

Project supported by Mingde Innovation Fund of Xi’an Mingde Institute of Technology (Project No. 2024MDE05).

Author contributions

Rolly R. Tang writing original draft preparation & methodology, Rolly R. Tang investigation & writing review and editing.

References

[1] Covi, E., Donati, E., Liang, X., Kappel, D., Heidari, H., Payvand, M. and Wang, W., 2021. Adaptive extreme edge computing for wearable devices. Frontiers in Neuroscience, 15, p.611300. https://doi.org/10.3389/fnins.2021.611300

[2] Bablu, T.A. and Rashid, M.T., 2025. Edge computing and its impact on real-time data processing for IoT-driven applications. Journal of Advanced Computing Systems, 5(1), pp.26–43. https://doi.org/10.69987/

[3] Chadwick, D.W., Fan, W., Costantino, G., De Lemos, R., Di Cerbo, F., Herwono, I., Manea, M., Mori, P., Sajjad, A. and Wang, X.S., 2020. A cloud-edge based data security architecture for sharing and analysing cyber threat information. Future Generation Computer Systems, 102, pp.710–722. https://doi.org/10.1016/j.future.2019.06.026

[4] Zhang, J., Qu, Z., Chen, C., Wang, H., Zhan, Y., Ye, B. and Guo, S., 2021. Edge learning: The enabling technology for distributed big data analytics in the edge. ACM Computing Surveys (CSUR), 54(7), pp.1-36. https://doi.org/10.1145/3464419

[5] Rahman, A., Hossain, M.S., Muhammad, G., Kundu, D., Debnath, T., Rahman, M., Khan, M.S.I., Tiwari, P. and Band, S.S., 2023. Federated learning-based AI approaches in smart healthcare: concepts, taxonomies, challenges, and open issues. Cluster computing, 26(4), pp.2271-2311. https://doi.org/10.1007/s10586-022-03658-4

[6] Ibrahim Khalaf, O., Algburi, S., S, A., Selvaraj, D., Sharif, M.S. and Elmedany, W., 2024. Federated learning with hybrid differential privacy for secure and reliable cross-IoT platform knowledge sharing. Security and Privacy, 7(3), p.e374. https://doi.org/10.1002/spy2.374

[7] Thantharate, P., Bhojwani, S. and Thantharate, A., 2024. DPShield: Optimizing differential privacy for high-utility data analysis in sensitive domains. Electronics, 13(12), p.2333. https://doi.org/10.3390/electronics13122333

[8] Chen, C., Liu, J., Tan, H., Li, X., Wang, K.I.K., Li, P., Sakurai, K. and Dou, D., 2025. Trustworthy federated learning: privacy, security, and beyond. Knowledge and Information Systems, 67(3), pp.2321-2356. https://doi.org/10.1007/s10115-024-02285-2

[9] Nguyen, D.C., Ding, M., Pham, Q.V., Pathirana, P.N., Le, L.B., Seneviratne, A., Li, J., Niyato, D. and Poor, H.V., 2021. Federated learning meets blockchain in edge computing: Opportunities and challenges. IEEE Internet of Things Journal, 8(16), pp.12806-12825. https://doi.org/10.1109/JIOT.2021.3072611

[10] Ahmed, A., Abdullah, S., Bukhsh, M., Ahmad, I. and Mushtaq, Z., 2022. An energy-efficient data aggregation mechanism for IoT secured by blockchain. IEEE Access, 10, pp.11404–11419. https://doi.org/10.1109/ACCESS.2022.3146295

[11] Nazir, A., He, J., Zhu, N., Anwar, M.S. and Pathan, M.S., 2024. Enhancing IoT security: a collaborative framework integrating federated learning, dense neural networks, and blockchain. Cluster Computing, 27(6), pp.8367-8392. https://doi.org/10.1007/s10586-024-04436-0

[12] Xu, W., Yang, Z., Ng, D.W.K., Levorato, M., Eldar, Y.C. and Debbah, M., 2023. Edge learning for B5G networks with distributed signal processing: Semantic communication, edge computing, and wireless sensing. IEEE journal of Selected Topics in Signal Processing, 17(1), pp.9-39. https://doi.org/10.1109/JSTSP.2023.3239189

[13] Shenoy, D., Bhat, R. and Krishna Prakasha, K., 2025. Exploring privacy mechanisms and metrics in federated learning. Artificial Intelligence Review, 58(8), p.223. https://doi.org/10.1007/s10462-025-11170-5

[14] Liu, W., Xu, X., Li, D., Qi, L., Dai, F., Dou, W. and Ni, Q., 2022. Privacy preservation for federated learning with robust aggregation in edge computing. IEEE Internet of Things Journal, 10(8), pp.7343-7355. https://doi.org/10.1109/JIOT.2022.3229122

[15] Jiang, B., Li, J., Wang, H. and Song, H., 2021. Privacy-preserving federated learning for industrial edge computing via hybrid differential privacy and adaptive compression. IEEE Transactions on Industrial Informatics, 19(2), pp.1136-1144. https://doi.org/10.1109/TII.2021.3131175

[16] Jia, B., Zhang, X., Liu, J., Zhang, Y., Huang, K., and Liang, Y., 2021. Blockchain-enabled federated learning data protection aggregation scheme with differential privacy and homomorphic encryption in IIoT. IEEE Transactions on Industrial Informatics, 18(6), pp.4049-4058. https://doi.org/10.1109/TII.2021.3085960

[17] Sharma, J., Kim, D., Lee, A. and Seo, D., 2021. On a differential privacy-based framework for enhancing user data privacy in a mobile edge computing environment. Ieee access, 9, pp.38107-38118. https://doi.org/10.1109/ACCESS.2021.3063603

[18] Wang, R., Lai, J., Zhang, Z., Li, X., Vijayakumar, P. and Karuppiah, M., 2022. Privacy-preserving federated learning for internet of Medical Things under edge computing. IEEE Journal of biomedical and health informatics, 27(2), pp.854-865. https://doi.org/10.1109/JBHI.2022.3157725

[19] Zhou, J., Wu, N., Wang, Y., Gu, S., Cao, Z., Dong, X. and Choo, K.K.R., 2022. A differentially private federated learning model against poisoning attacks in edge computing. IEEE Transactions on Dependable and Secure Computing, 20(3), pp.1941-1958. https://doi.org/10.1109/TDSC.2022.3168556

[20] Zhou, H., Yang, G., Dai, H. and Liu, G., 2022. PFLF: Privacy-preserving federated learning framework for edge computing. IEEE Transactions on Information Forensics and Security, 17, pp.19051905-1918. https://doi.org/10.1109/TIFS.2022.3174394
[21] Hongbin, F. and Zhi, Z., 2023. Privacy-preserving data aggregation scheme based on federated learning for IIoT. Mathematics, 11(1), p.214. https://doi.org/10.3390/math11010214
[22] Liu, Y., Liu, P., Jing, W. and Song, H.H., 2023. PD2S: A privacy-preserving differentiated data sharing scheme based on blockchain and federated learning. IEEE Internet of Things Journal, 10(24), pp.21489-21501. https://doi.org/10.1109/JIOT.2023.3295763
[23] Wu, X., Zhang, Y., Shi, M., Li, P., Li, R. and Xiong, N.N., 2022. An adaptive federated learning scheme with differential privacy preserving. Future Generation Computer Systems, 127, pp.362-372. https://doi.org/10.1016/j.future.2021.09.015
[24] Fan, H., Huang, C. and Liu, Y., 2022. Federated learning-based privacy-preserving data aggregation scheme for IIoT. IEEE Access, 11, pp.6700-6707. http://dx.doi.org/10.13140/RG.2.2.16605.59365
[25] Zhu, B. and Niu, L., 2025. A privacy-preserving federated learning scheme with homomorphic en cryption and edge computing. Alexandria Engineering Journal, 118, pp.11-20.
[26] Almalki, F.A. and Soufiene, B.O., 2021. EPPDA: An efficient and privacy-preserving data aggregation scheme with authentication and authorization for IoT-based healthcare applications. Wireless Communications and Mobile Computing, 2021(1), p.5594159. https://doi.org/10.1155/2021/5594159
[27] [27] Zhang, Z., Yang, W., Wu, F. and Li, P., 2023. Privacy and integrity-preserving data aggregation scheme for wireless sensor networks digital twins. Journal of Cloud Computing, 12(1), p.140. https://doi.org/10.1186/s13677-023-00522-7
[28] [28] Alabdulatif, A., 2025. GuardianAI: Privacy-preserving federated anomaly detection with differential privacy. Array, 26, p.100381. https://doi.org/10.1016/j.array.2025.100381
[29] [29] Reis, M.J. and Serôdio, C., 2025. Edge AI for real-time anomaly detection in smart homes. Future Internet, 17(4), p.179. https://doi.org/10.3390/fi17040179