

Article

Algorithm Design and Analysis of a Blockchain-Enabled Reinforcement and Active Learning Framework for Robust IoT Intrusion Detection

Nadeem Javaid 

Computer Science Department, Faculty of Computers and Information Technology, University of Tabuk, Tabuk 71491, Saudi Arabia; njavid@ut.edu.sa

Abstract

The fast development of Internet of Things (IoT) gadgets has brought both a new level of connectivity and automation, at the same time making the networks more vulnerable to more sophisticated cyber attacks. Existing intrusion detection systems have a number of fundamental limitations, such as extreme class imbalance in network traffic data, inadequate feature interaction modeling, inability to train on dynamic attack patterns, heavy reliance on fully labeled data, weak evaluation capabilities, and limited interpretability. To deal with these issues, this research employs a data balancing mechanism with an autoencoder to reduce class imbalance by learning meaningful latent representations of the minority attack classes. A Parallel Hybrid single-step BiLSTM-FCNN (PH-BiLSTM-FCNN) model is then proposed as a single network to learn contextual dependencies and nonlinear discriminative features simultaneously along parallel learning pathways. To further enhance adaptability and convergence stability, a Q-learning Optimized PH-BiLSTM-FCNN (Q-PH-BiLSTM-FCNN) is introduced, enabling dynamic optimization of training behavior based on feedback-driven interactions. In addition, an Entropy-based Active Learning on PH-BiLSTM-FCNN (EAL-PH-BiLSTM-FCNN) is developed to significantly reduce labeling requirements by selectively querying the most informative samples. Furthermore, a blockchain layer with smart contracts is embedded consistently in all of the proposed models to ensure tamper-resistant logging, transparent validation, and reliable recording of training and evaluation results. From an algorithm design perspective, the proposed framework is structured as a set of verifiable procedures for PH-BiLSTM-FCNN training, Q-learning optimization, and entropy-based active sample selection. Its algorithmic behavior and robustness are evaluated through execution-time analysis, 10-fold cross-validation, and permutation feature importance, ensuring both predictive reliability and interpretability. Experimental results demonstrate that the proposed PH-BiLSTM-FCNN, Q-PH-BiLSTM-FCNN, and EAL-PH-BiLSTM-FCNN models consistently outperform state-of-the-art FCNN, Bi-LSTM, GRU, LSTM, LR models, achieving improvements of 4.80%, 13.52%, and 6.85% in accuracy, 4.81%, 13.53%, 6.86% in recall, and 5.18%, 18.44%, and 8.52% in precision recall-area under the curve, respectively. These results affirm that the proposed framework provides improved detection performance, statistical reliability, and interpretability while addressing practical deployment limitations in IoT security settings.



Academic Editors: Chih-Lung Lin and Chi-Hung Chuang

Received: 13 July 2026

Revised: 10 September 2026

Accepted: 14 September 2026

Published: 19 September 2026

Copyright: © 2026 by the author.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and

conditions of the [Creative Commons](#)

[Attribution \(CC BY\)](#) license.

Keywords: Internet of Things; intrusion detection; autoencoder; fully connected neural network; entropy-based active learning; Q learning; 10-fold cross validation; permutation feature importance

1. Introduction

The Internet of Things (IoT) allows connecting heterogeneous devices at scale, including sensors, actuators, smart appliances, industrial controllers, and providing intelligent services in such domains as smart cities, healthcare, transportation, and industrial automation [1–3]. The open and distributed qualities of IoT environments, coupled with resource limitations and the heterogeneity of protocols, pose serious security threats [4]. IoT devices are usually used on open networks, and in unmonitored environments, thus allowing them to become more vulnerable to unlawful operations and attacks by cybercriminals and intruders [5].

A range of cyber attacks are also usually targeted at IoT networks, including Denial-of-Service (DoS) attacks [6] that disrupt network or device availability by draining network or device resources, reconnaissance and probing attacks that find sensitive information regarding network structure, backdoor attacks that provide unauthorized access persistently, and command injection attacks that enable adversaries to remotely control compromised devices. The attacks lead to complicated and changing traffic flows that tend to disorient normal traffic. The presence of several types of attacks, as well as extremely uneven traffic distributions, complicates the process of proper detection in the conditions of the real-world IoT implementation.

The Intelligent Detection Systems (IDSs) have an important role in overseeing IoT network traffic and detecting any malicious activity by intelligently analyzing data flows. The IDS models based on Machine Learning (ML) and Deep Learning (DL) have shown a high ability to learn nonlinear feature representations and subtle attack patterns in high-dimensional traffic data [7]. State-of-the-art architectures that include dense learning of features and attention can be used to model global feature interactions and context dependencies and offer reliable multi-class intrusion detection in dynamic and heterogeneous IoT settings.

The rest of this paper is structured in the following way. Section 2 discusses literature related work and identifies the research gaps. Section 3 outlines the proposed system model, such as data preprocessing, balancing strategy, and the architecture of the proposed models. Section 4 provides the experimental design, analysis of performance, and the comparison of the results to the cross-validation and statistical tests. Lastly, Section 5 is a conclusion with potential future work directions.

2. Related Work

Recent research has investigated approaches such as hybrid DL, ensemble learning and active learning methods in order to enhance the performance of intrusion detection in IoT and Industrial Internet of Things (IIoT) setups. In ref. [8], the authors of the study present a hybrid Convolutional Neural Network-single-step Bi-Directional Long Short Term Memory (CNN-Bi-LSTM) integrated intrusion detection framework with federated learning to maintain data privacy in IIoT systems. The CNN component effectively captures spatial features from network traffic, while the single-step Bi-LSTM models bidirectional feature dependencies. Nevertheless, the major drawback of LSTM-based models is that they incur a great amount of computational and memory overhead that makes them not so suitable to apply in real-time in resource-constrained IoT and IIoT devices, particularly when paired with federated learning communication expenses.

In ref. [9], the authors propose a hybrid CNN-Gated Recurrent Unit (CNN-GRU) architecture to detect intrusion, which is tested on the IoTID20 and BoT-IoT datasets. The model uses CNNs to extract features and GRUs to learn in sequence, in addition to correcting the imbalance in classes in large datasets. GRU networks are computationally lightweight compared to LSTMs, but one significant limitation is that they are less sensitive

to much longer time dependencies, potentially resulting in suboptimal detection of more complex or subtle attack patterns with long time horizons.

The authors in ref. [10] introduce a Logistic Regression-based Ensemble Classifier (LREC) combining Adaptive Boosting (AdaBoost) and Random Forest (RF) classifiers with Adaptive Synthetic Sampling (ADASYN) to address data imbalance and recursive feature elimination to address feature selection. Although the proposed ensemble enhances interpretability and robustness, the primary weakness of LR-based models is that they assume the decision boundary between normal and attack activities is linear, which limits these models to modeling highly non-linear and complex attack activities, which are common in modern IoT intrusion data sets.

In ref. [11] a thorough benchmarking of the pool-based active learning strategies is performed, such as Uncertainty-based Active Learning (UAL) sampling, ranked batch-mode sampling, and query by committee. Though UAL is less costly in terms of labeling, since it chooses informative samples, its core drawback is that it favors ambiguous or noisy examples and this may result in suboptimal model updates and poor generalization, especially when using highly imbalanced intrusion detection data.

In ref. [12], the authors use a generative artificial intelligence system, which is the Conditional Tabular Generative Adversarial Network (CTGAN) to produce artificial data that can be used to enhance performance in intrusion detection. CTGAN has some limitations in spite of its effectiveness in the modeling of complex tabular data distributions. Specifically, it is susceptible to training instability and mode collapse, which may lead to the lack of diversity among the generated samples. Furthermore, CTGAN can enhance noise and bias that exist within the original dataset, resulting in unrealistic or poor quality synthetic examples that can have a harmful impact on the generalization capacity of the downstream intrusion detection model.

In ref. [13], optimized CNN-based IDS, which considers the issue of data imbalance, is proposed by the authors to be applied in IIoT settings and used to resolve the issue of data imbalance with the help of the multi-class version of the Synthetic Minority Over-sampling Technique (SMOTE). Though SMOTE is good at balancing the classes, its main weakness is that it creates synthetic samples by linearly interpolating between the minority class samples and the actual underlying data distribution. It can result in overlapping of classes and introduction of ambiguous samples, especially in high-dimension feature spaces, which can further raise the chances of overfitting and lowering the resilience of the model to unknown attack patterns.

The authors combine CNN, LSTM networks, and Reinforcement Learning (RL) models, such as Deep Q-Network (DQN) and Proximal Policy Optimization (PPO), to enhance the detection of changing cyber threats in ref. [14]. Despite the fact that PPO is characterized by stable and consistent updates of its policies, one central shortcoming of PPO is that it has a high sample complexity, necessitating a large number of interactions with the environment to reach optimal performance. This renders PPO less effective in data-sparse or simulationally-intensive intrusion detection systems, and can be counterproductive to the flexibility of PPO to evolving attack patterns in real-time systems.

The authors in ref. [15] suggest a Deep RL (DRL)-based framework to improve the performance of network IDS with various DRL algorithms. The methods used are Deep Q-Network (DQN), Deep Deterministic Policy Gradient (DDPG), PPO, Twin Delayed Deep Deterministic Policy Gradient (TD3), and Soft Actor-Critic (SAC). One of them is that DDPG is sensitive to hyperparameter tuning and exploration noise, and thus, typically, training becomes unstable and inefficient in complex or non-stationary tasks. Such sensitivity may lead to poor policies and less strength when used in dynamic and high-dimensional problems of intrusion detection.

2.1. Research Gaps

Although there are significant improvements in intrusion detection in IoT, some important research gaps exist. First, recent studies still tend to use extremely unbalanced datasets, with the prevalent normal traffic overwhelming classes of minority attacks, and thus biasing the learning process and failing to detect any sign of rare and important intrusion. Second, the current methods usually cannot combine complementary learning behaviors into a single framework because most of the existing models emphasize one perspective of the representation, hindering their capability to address the complexity, heterogeneity, and nonlinear nature of the IoT traffic. Third, current works have very little flexibility in their training strategies, as they are mostly fixed and predetermined, with no adaptive mechanisms that respond to changing traffic patterns and continually update the learning behavior with performance feedback. Further most of these frameworks also assume full labeling of the data, which, although not a realistic assumption in practical real-world deployments, is expensive and time-consuming to achieve, leading to an inefficient utilization of labeling resources. Forth, the assessment procedures in most research studies are still weak and cross-validation across fold is not well used to determine reliability in generalization and strength under varying data partitions. Fifth, the testing of statistical significance is frequently not performed, and any reported performance improvement is not backed by rigorous evidence, which diminishes the validity of comparative tests. Lastly, interpretability can often be neglected as most successful models are presented as black boxes where their predictions are not clearly explained, which results in lack of trust, inability validating by an expert, and limits their practical use in IoT settings where their predictions are vital to security.

2.2. Contributions

There are several important contributions to this study which aim at enhancing intrusion detection in IoT environments. First, an autoencoder-based data balancing approach is implemented to reliably overcome severe imbalance in the classes by training compact latent representations of minority classes and producing realistic synthetic samples, which preserves the inherent data distributions and increases model learning across the entire types of attacks. Second, a Parallel Hybrid single-step BiLSTM-FCNN (PH-BiLSTM-FCNN) model is introduced, which uses a single-step Bi-directional Long-Short Term Memory (BiLSTM) nonlinear encoder and a Fully Connected Neural Network (FCNN) in a parallel structure to jointly learn feature dependency and nonlinear discriminative patterns, leading to more robust and precise multi-class intrusion detection. Third, Q-learning-Optimized PH-BiLSTM-FCNN (Q-PH-BiLSTM-FCNN) model is proposed to optimize the training dynamics adaptively through the interaction-based feedback of learning the best hyperparameter configurations, resulting in better convergence stability and better detection performance in the presence of heterogeneous traffic. Fourth, an Entropy-based Active Learning on PH-BiLSTM-FCNN (EAL-PH-BiLSTM-FCNN) is proposed, which minimizes labeling overhead by actively sampling the most informative and uncertain samples, thus providing competitive performance using significantly fewer labeled examples. Moreover, a mechanism of smart contract, supported by blockchain, is implemented consistently in all of the models proposed to enable tamper-resistance logging, transparent validation, and reliable tracking of model configurations and performance measures. Also, 10-Fold Cross Validation (10-FCV) is implemented in order to critically test the model robustness and generalization so that the performance of the model is stable under various data partitions. A Nadeau–Bengio dependence-corrected resampled *t*-test, together with Holm-adjusted *p*-values, corrected 95% confidence intervals, and effect-size analysis, is used to assess the reliability of the observed performance differences while accounting for dependence

among the overlapping cross-validation folds. Lastly, the Permutation Feature Importance (PFI) analysis is performed to improve interpretability by estimating how much a single feature influences predictions made by the model, thus making it possible to make decisions transparently and analyzing intrusion-related behavior by experts.

The individual components employed in the proposed framework, including autoencoder-based balancing, FCNNs, bidirectional long short-term memory networks, nonlinear feature transformation mechanisms, Q-learning, EAL, and blockchain-assisted logging, are established techniques. Therefore, the novelty of this study is not claimed at the level of these individual components. Instead, the principal contribution lies in their coordinated and task-specific integration into a unified blockchain-enabled IIoT intrusion detection framework. The proposed framework jointly addresses class imbalance through class-specific latent augmentation, nonlinear traffic representation through parallel recurrent and fully connected branches, adaptive hyperparameter selection through Q-learning, limited label availability through EAL, and result traceability through blockchain and smart-contract-assisted logging.

3. Proposed System Model

3.1. Algorithm Design and Analysis

The proposed framework is designed as an algorithmically structured intrusion detection system rather than a single black-box classifier. Algorithm 1 defines the PH-BiLSTM-FCNN training process with blockchain-assisted validation, Algorithm 2 extends this design through discrete Q-learning-based action selection, and Algorithm 3 introduces entropy-driven active sample querying. For N training samples, feature dimension d , batch size B , number of epochs E , hidden dimension h , Q-learning episodes E_Q , active learning iterations T_A , and number of classes C , the dominant PH-BiLSTM-FCNN training complexity can be expressed as

$$\mathcal{O}\left(E \left\lceil \frac{N}{B} \right\rceil (C_{\text{FCNN}} + C_{\text{PN}})\right), \quad (1)$$

where

$$C_{\text{FCNN}} = \sum_{\ell=1}^L d_{\ell-1} d_{\ell}, \quad C_{\text{PN}} = \mathcal{O}(dh^2 + dh). \quad (2)$$

Here, C_{FCNN} represents the dense-layer computation in the FCNN branch, while C_{PN} denotes the recurrent and nonlinear feature transformation computation in the single-step BiLSTM nonlinear encoder branch. The Q-PH-BiLSTM-FCNN module adds $\mathcal{O}(E_Q)$ Q-table update operations over the discrete action space, whereas EAL-PH-BiLSTM-FCNN adds $\mathcal{O}(T_A | X_U C)$ entropy-scoring operations over the unlabeled pool X_U . This analysis clarifies the computational role of each algorithmic component and supports a fair interpretation of the accuracy, robustness, and execution-time trade-offs reported in the experimental Section 4.

Algorithm 1 Detailed Algorithm of the Proposed PH-BiLSTM-FCNN

Require: Training data (X_{tr}, y_{tr}) , validation data (X_{val}, y_{val}) , untouched test data (X_{test}, y_{test}) , epochs E , and batch size B

Ensure: Trained PH-BiLSTM-FCNN model $\mathcal{M}$ with auditable final evaluation record

- 1: Set random seeds for reproducibility
- 2: Convert training labels into categorical representation
- 3: Initialize FCNN, single-step BiLSTM, decoder-LSTM, fusion, and classifier parameters
- 4: Initialize blockchain ledger $\mathcal{B}$ with a genesis block
- 5: **for** epoch = 1 to E **do**
- 6: **for** each mini-batch (X_b, Y_b) **do**
- 7: Compute FCNN representation: $\mathbf{f}_{FCNN} \leftarrow \text{FCNN}(X_b)$
- 8: Reshape traffic observations: $\mathbf{X}_r \leftarrow \text{reshape}(X_b, B, 1, d)$
- 9: Compute bidirectional representation: $\mathbf{H} \leftarrow \text{single-stepBiLSTM}(\mathbf{X}_r)$
- 10: Compute mean recurrent state $\mathbf{s}$
- 11: Compute the single-position normalized coefficient $\alpha_1 = 1$
- 12: Obtain context representation: $\mathbf{c} \leftarrow \mathbf{H}_1$
- 13: Compute decoder representation: $\mathbf{d} \leftarrow \text{LSTM}_{dec}(\mathbf{c})$
- 14: Compute nonlinear recurrent representation: $\mathbf{r} \leftarrow \tanh(\mathbf{W}_r \mathbf{d} + \mathbf{b}_r)$
- 15: Compute recurrent-branch class representation: $\mathbf{p}_r \leftarrow \text{softmax}(\mathbf{W}_p \mathbf{r} + \mathbf{b}_p)$
- 16: Fuse both representations: $\mathbf{f}_{fusion} \leftarrow [\mathbf{f}_{FCNN} \parallel \mathbf{p}_r]$
- 17: Compute fusion representation: $\mathbf{h}_{fusion} \leftarrow \tanh(\mathbf{W}_g \mathbf{f}_{fusion} + \mathbf{b}_g)$
- 18: Compute final prediction: $\hat{Y} \leftarrow \text{softmax}(\mathbf{W}_c \mathbf{h}_{fusion} + \mathbf{b}_c)$
- 19: Compute categorical cross-entropy loss $\mathcal{L}_{CE}$
- 20: Update all trainable parameters using SGD
- 21: **end for**
- 22: Monitor training behavior using the validation partition
- 23: **end for**
- 24: Fix the final PH-BiLSTM-FCNN model configuration
- 25: Evaluate the finalized model once on the untouched test set (X_{test}, y_{test})
- 26: Compute final held-out test performance metrics
- 27: Record authorized model metadata and final evaluation information in blockchain $\mathcal{B}$
- 28: Verify blockchain integrity
- 29: **Return** finalized PH-BiLSTM-FCNN model $\mathcal{M}$

3.2. Dataset Description

The WUSTL-IIoT-2021 dataset, developed by Washington University in St. Louis, is intended to support research on intrusion detection in IIoT environments [16]. The dataset is collected in a controlled testbed which mimics real industrial control systems and benign as well as malicious network traffic. It is specifically beneficial in the analysis of attacks that are stealthy, protocol-specific, and have the ability to hit the critical infrastructure components. This dataset contains approximately 1,194,464 samples divided into five main classes: normal, DoS, backdoor, reconnaissance and command injection. In each instance, there are 49 statistical and temporal features that have been extracted based on network traffic flows, which capture packet-level and flow-level behavior. These characteristics involve the number of sent packets, the number of sent and received bytes, the duration of flows, and the period between arrivals. This dataset is provided in CSV format and it is fully annotated with supervised learning activities and sequences can be built by using timestamps.

Algorithm 2 Detailed Algorithm of State-Dependent Q-PH-BiLSTM-FCNN

```

1: Input: Training data  $(X_{tr}, y_{tr})$ , fixed validation data  $(X_{val}, y_{val})$ , untouched test data  $(X_{test}, y_{test})$ , action space  $\mathcal{A}$ , and number of episodes  $E$ 
Ensure: Optimized Q-PH-BiLSTM-FCNN model with auditable optimization history
2: Initialize  $Q(s, a) = 0$  for all admissible state-action pairs
3: Set Q-learning parameters  $\alpha$ ,  $\gamma$ , and  $\epsilon$ 
4: Convert training and validation labels to categorical representation
5: Initialize blockchain ledger  $\mathcal{B}$  with genesis block
6: Deploy smart contract  $\mathcal{C}$  for episode authorization
7: Initialize the optimization state  $s_1$  from the initial validation status
8: for episode  $t = 1$  to  $E$  do
9:   Observe the current optimization state  $s_t$ 
10:  Select action  $a_t$  using the  $\epsilon$ -greedy policy over  $Q(s_t, a)$ 
11:  Configure PH-BiLSTM-FCNN using the learning rate and batch size specified by  $a_t$ 
12:  Train PH-BiLSTM-FCNN exclusively on  $(X_{tr}, y_{tr})$  for the predefined number of epochs
13:  Evaluate the trained model exclusively on the fixed validation set  $(X_{val}, y_{val})$ 
14:  Compute current validation performance  $P_t$ 
15:  Compute performance variation  $\Delta P_t = P_t - P_{t-1}$ 
16:  Compute validation predictive uncertainty  $U_t$ 
17:  Compute optimization progress  $G_t = t/E$ 
18:  Construct the next optimization state  $s_{t+1}$ 
19:  Compute validation-based reward  $r_t$ 
20:  Update  $Q(s_t, a_t)$  using  $r_t + \gamma \max_{a'} Q(s_{t+1}, a')$ 
21:  if  $\mathcal{C}$  authorizes  $r_t$  then
22:    Append  $s_t, a_t, r_t$ , validation metrics, selected hyperparameters, and execution metadata to blockchain  $\mathcal{B}$ 
23:  end if
24:  Save model checkpoint
25: end for
26: Select the optimal hyperparameter configuration from the completed state-action optimization trajectory
27: Initialize the final Q-PH-BiLSTM-FCNN using the selected optimal hyperparameters
28: Train the final Q-PH-BiLSTM-FCNN on  $(X_{tr}, y_{tr})$ 
29: Fix the final model configuration before accessing the held-out test partition
30: Evaluate the finalized Q-PH-BiLSTM-FCNN once on the untouched test set  $(X_{test}, y_{test})$ 
31: Compute the final held-out test-set performance metrics
32: Record the final test results separately in blockchain  $\mathcal{B}$ 
33: Verify blockchain integrity
34: Return optimized Q-PH-BiLSTM-FCNN model

```

3.3. Preprocessing

The preprocessing phase intends to maintain consistency of data, eliminate redundancy, and make the dataset ready to be trained on a reliable model. First, the dataset is analyzed with the help of null-value inspection functions, and no serious inconsistencies are found. The redundant records are then detected and deleted to remove the redundant samples that may affect the learning process. The target variable Traffic exhibits a highly imbalanced multi-class distribution, consisting of 1,107,448 samples labeled as Normal, 78,305 as DoS, 8240 as Recon, 259 as CommInj, and 212 as Backdoor. Since the dataset contains categorical attributes, label encoding is applied to transform categorical values into numerical representations [17] suitable for ML models. For a given categorical feature X with k unique categories $\{c_1, c_2, \dots, c_k\}$, label encoding maps each category to an integer label according to the function:

$$\phi(c_i) = i - 1, \quad \forall i \in \{1, 2, \dots, k\}, \quad (3)$$

where $\phi(c_i)$ denotes the encoded numerical value corresponding to category c_i . Prior to encoding, missing categorical values are handled by assigning a placeholder category to preserve data integrity. After encoding, the feature matrix X and target vector y are separated, and a stratified sampling strategy is employed to extract a representative subset of 150,000 instances while maintaining the original class proportions. This stratified subset is then used for subsequent training and evaluation, ensuring that minority attack classes are proportionally preserved during model learning.

Algorithm 3 Detailed Algorithm of EAL-PH-BiLSTM-FCNN

- 1: **Input:** Initial labeled set $X_L^{(0)}$, unlabeled pool $X_U^{(0)}$, validation set $(X_{\text{val}}, y_{\text{val}})$, untouched test set $(X_{\text{test}}, y_{\text{test}})$, iterations T , query size Q , epochs E , batch size B , entropy threshold τ
 - 2: Initialize blockchain ledger $\mathcal{B}$ with genesis block
 - 3: Deploy smart contract $\mathcal{C}$ with minimum entropy threshold τ
 - 4: **for** $t = 1$ to T **do**
 - 5: Convert labels in $X_L^{(t-1)}$ to categorical representation
 - 6: Train PH-BiLSTM-FCNN on $X_L^{(t-1)}$ for E epochs with batch size B
 - 7: Evaluate the model on $(X_{\text{val}}, y_{\text{val}})$ and compute iteration-level validation metrics
 - 8: Predict class probabilities for all $x \in X_U^{(t-1)}$
 - 9: Compute entropy $H(x)$ for each unlabeled sample
 - 10: Obtain smart-contract-authorized candidates $\mathcal{A}_t \leftarrow \mathcal{C}(H(x) \geq \tau)$
 - 11: Rank $\mathcal{A}_t$ by entropy and select at most the top- Q samples to form $\mathcal{Q}_t$
 - 12: Append validation metrics and query statistics to blockchain $\mathcal{B}$
 - 13: Update pools:
 - 14: $X_L^{(t)} \leftarrow X_L^{(t-1)} \cup \mathcal{Q}_t$
 - 15: $X_U^{(t)} \leftarrow X_U^{(t-1)} \setminus \mathcal{Q}_t$
 - 16: **end for**
 - 17: Fix the final labeling budget and enriched labeled set $X_L^{(T)}$
 - 18: Train/finalize EAL-PH-BiLSTM-FCNN using $X_L^{(T)}$
 - 19: Evaluate the finalized EAL-PH-BiLSTM-FCNN once on the untouched test set $(X_{\text{test}}, y_{\text{test}})$
 - 20: Compute and record the final test-set performance metrics separately in blockchain $\mathcal{B}$
 - 21: Verify blockchain integrity
 - 22: **Output:** Final EAL-PH-BiLSTM-FCNN model with auditable active-learning history and independent final test evaluation
-

3.4. Autoencoder-Based Data Balancing

To address the severe class imbalance present in the dataset, an autoencoder-based data augmentation strategy is employed to synthetically balance minority classes. Prior to balancing, all feature values are normalized using Min-Max scaling to map each feature into the range $[0, 1]$, which stabilizes neural network training and ensures uniform feature contribution [18]. Let $X \in \mathbb{R}^{n \times d}$ denote the normalized feature matrix with n samples and d features. The normalized dataset is then recombined with the target variable Traffic to facilitate class-wise processing.

An autoencoder architecture is designed to learn compact latent representations of each class independently [19]. The encoder function $f_\theta(\cdot)$ maps the input data to a lower-dimensional latent space, as Figure 1 shows, while the decoder function $g_\phi(\cdot)$ reconstructs the original input from this latent representation. Formally, for an input sample $\mathbf{x} \in \mathbb{R}^d$, the encoding and decoding processes are defined as

$$\mathbf{z} = f_\theta(\mathbf{x}), \quad \hat{\mathbf{x}} = g_\phi(\mathbf{z}), \quad (4)$$

where $\mathbf{z} \in \mathbb{R}^{d'}$ represents the latent embedding with $d' < d$. The autoencoder parameters θ and ϕ are optimized by minimizing the mean squared reconstruction error:

$$\mathcal{L}_{AE} = \frac{1}{n} \sum_{i=1}^n \|\mathbf{x}_i - \hat{\mathbf{x}}_i\|_2^2. \quad (5)$$

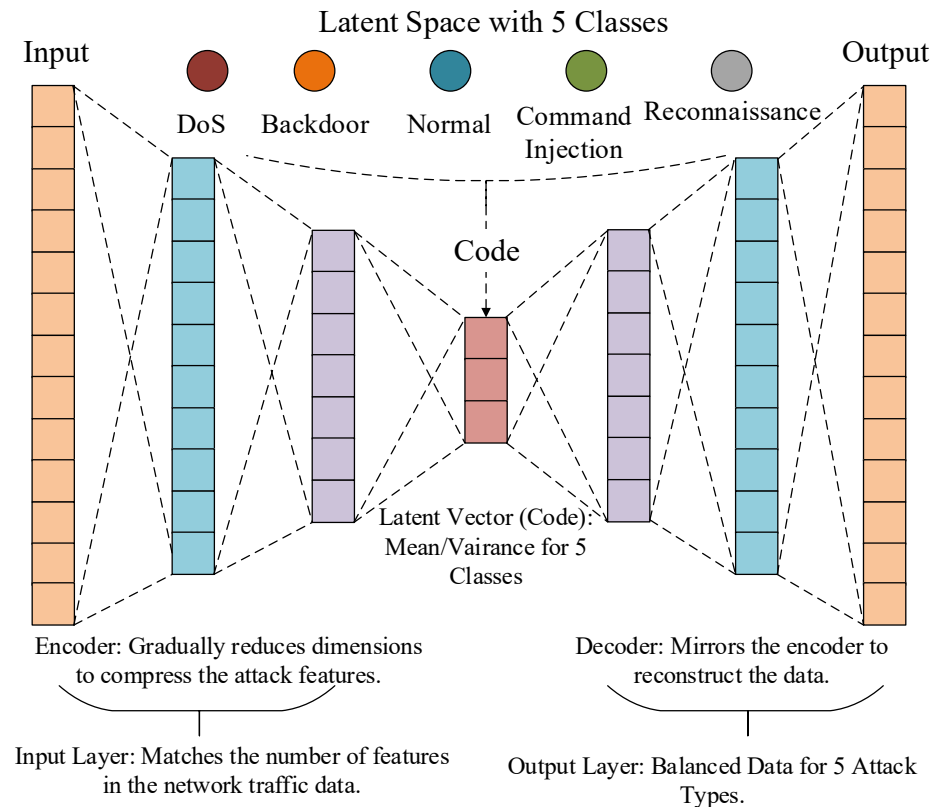


Figure 1. Working of autoencoder balancing technique.

To balance the dataset, a separate autoencoder is trained for each traffic class using only the samples belonging to that class. After training, the encoder is used to project the class samples into the latent space, and small Gaussian perturbations are added to these latent vectors to encourage diversity while preserving class-specific characteristics. This perturbation process is defined as

$$\tilde{\mathbf{z}} = \mathbf{z} + \epsilon, \quad \epsilon \sim \mathcal{N}(0, \sigma^2), \quad (6)$$

where σ controls the noise magnitude. The perturbed latent representations are then passed through the decoder to generate synthetic samples that follow the learned data distribution of the corresponding class. These synthetic samples are subsequently inverse-transformed to the original feature scale to maintain consistency with real observations.

The augmentation process is repeated until each minority class reaches the same number of samples as the original majority class. After generating synthetic instances for all underrepresented classes, the real and synthetic samples are merged and randomly shuffled to prevent ordering bias. As a result, the final balanced dataset contained an equal number of samples for all traffic categories, with each class consisting of 139,073 instances. This autoencoder-driven balancing strategy enables effective learning from highly imbalanced intrusion data while preserving intrinsic class structures and reducing overfitting associated with naive oversampling techniques. Following autoencoder-based balancing, the resulting dataset containing 695,365 records is randomly divided into

training and test partitions using an 80:20 stratified split. The preprocessing, autoencoder-based balancing, and subsequent 80:20 stratified splitting described above constitute the primary experimental pipeline used for the main comparative results of this study. To directly examine the potential data leakage, an additional leakage-controlled sensitivity experiment is conducted separately, in which the original real observations are partitioned before learned preprocessing and autoencoder-based augmentation, and synthetic samples are restricted to the training partition. This supplementary experiment is reported separately and is not intended to redefine the primary experimental pipeline.

The original dataset contained 1,194,464 traffic records, including 1,107,448 Normal, 78,305 DoS, 8240 Recon, 259 CommInj, and 212 Backdoor observations. A stratified subset of 150,000 records is subsequently extracted while approximately preserving the original class proportions. Consequently, the sampled dataset contained 139,073 Normal, 9833 DoS, 1035 Recon, 32 CommInj, and 27 Backdoor records. The Normal class is used as the balancing reference, and the remaining classes are synthetically expanded to 139,073 records each using the class-specific autoencoder. Table 1 reports the corresponding real and synthetic sample counts.

Table 1. Class-wise real and synthetic sample counts throughout the proportional sampling and autoencoder-based balancing stages.

Traffic Class	Original Full Dataset	Real Records in 150,000 Subset	Synthetic Records Generated	Final Balanced Records
Normal	1,107,448	139,073	0	139,073
DoS	78,305	9833	129,240	139,073
Recon	8240	1035	138,038	139,073
CommInj	259	32	139,041	139,073
Backdoor	212	27	139,046	139,073
Total	1,194,464	150,000	545,365	695,365

3.5. Parallel Hybrid Single-Step BiLSTM-FCNN (PH-BiLSTM-FCNN) Model

The proposed PH-BiLSTM-FCNN is designed for multi-class network traffic classification by integrating two independently parameterized nonlinear learning pathways. The first pathway employs a FCNN to learn a hierarchical nonlinear representation directly from the tabular traffic vector, whereas the second pathway employs a single-step BiLSTM-based nonlinear transformation of the complete traffic observation. The outputs of the two pathways are subsequently concatenated and processed by a trainable fusion classifier. Let a mini-batch of network traffic observations be represented as

$$\mathbf{X} \in \mathbb{R}^{B \times d}, \quad (7)$$

where B denotes the mini-batch size and d denotes the number of predictor variables.

The first pathway serves as the primary FCNN feature-learning branch. For the l -th dense transformation, its hidden representation is expressed as

$$\mathbf{h}_f^{(l)} = \tanh\left(\mathbf{W}_f^{(l)} \mathbf{h}_f^{(l-1)} + \mathbf{b}_f^{(l)}\right), \quad (8)$$

where $\mathbf{W}_f^{(l)}$ and $\mathbf{b}_f^{(l)}$ denote the trainable weight matrix and bias vector, respectively. The FCNN branch consists of dense layers containing 128, 64, and 32 neurons with tanh activation. Batch normalization and dropout with a rate of 0.2 are applied after the 128- and 64-neuron layers to stabilize optimization and reduce overfitting. The resulting feature representation is

$$\mathbf{f}_{\text{FCNN}} \in \mathbb{R}^{B \times 32}. \quad (9)$$

The second pathway performs an additional nonlinear transformation using recurrent units. In accordance with the implemented architecture, each d -dimensional traffic observation is reshaped as a single sequence position:

$$\mathbf{X}_r = \text{reshape}(\mathbf{X}) \in \mathbb{R}^{B \times 1 \times d}. \quad (10)$$

Therefore, the complete set of d predictor variables forms one sequence element rather than d independent temporal or feature positions.

The reshaped observation is processed by a single-step Bi-LSTM containing $h = 64$ hidden units in each direction:

$$\mathbf{H} = \text{BiLSTM}(\mathbf{X}_r), \quad (11)$$

where

$$\mathbf{H} \in \mathbb{R}^{B \times 1 \times 2h}. \quad (12)$$

For $h = 64$, the bidirectional representation at the single sequence position contains 128 latent values.

The implementation subsequently computes a normalized score over the sequence dimension. For a general sequence length L , the normalized coefficient can be expressed as

$$\alpha_t = \frac{\exp(e_t)}{\sum_{j=1}^L \exp(e_j)}, \quad (13)$$

where

$$e_t = \mathbf{v}^T \tanh(\mathbf{W}_1 \mathbf{H}_t + \mathbf{W}_2 \mathbf{s}), \quad (14)$$

and $\mathbf{s}$ denotes the mean encoded representation.

However, the implemented input contains only one sequence position, i.e., $L = 1$. Consequently

$$\alpha_1 = \frac{\exp(e_1)}{\exp(e_1)} = 1. \quad (15)$$

The corresponding context representation therefore reduces to

$$\mathbf{c} = \sum_{t=1}^1 \alpha_t \mathbf{H}_t = \mathbf{H}_1. \quad (16)$$

Accordingly, this normalized weighting operation does not perform feature-level attention, temporal attention, or relevance ranking among predictor variables. It is retained as part of the implemented nonlinear transformation pathway, but no architectural novelty is attributed to the weighting operation itself. The recurrent pathway should therefore be interpreted as a single-step BiLSTM-based nonlinear representation of the complete tabular observation.

The resulting context vector is provided to a decoder LSTM containing 64 hidden units:

$$\mathbf{d} = \text{LSTM}_{\text{dec}}(\mathbf{c}), \quad (17)$$

where

$$\mathbf{d} \in \mathbb{R}^{B \times 64}. \quad (18)$$

The decoder representation is subsequently transformed through a dense layer with tanh activation:

$$\mathbf{r} = \tanh(\mathbf{W}_r \mathbf{d} + \mathbf{b}_r), \quad (19)$$

followed by a C-class softmax transformation:

$$\mathbf{p}_r = \text{softmax}(\mathbf{W}_p \mathbf{r} + \mathbf{b}_p), \quad (20)$$

where:

$$\mathbf{p}_r \in \mathbb{R}^{B \times C}. \quad (21)$$

The recurrent branch therefore provides an independently learned nonlinear class representation that complements the 32-dimensional representation generated by the FCNN branch. The purpose of combining the two pathways is to exploit heterogeneous nonlinear transformations rather than to claim explicit temporal or feature-level attention.

As illustrated in Figure 2, the representation generated by the FCNN branch and the output representation generated by the single-step BiLSTM-based nonlinear branch are combined through concatenation:

$$\mathbf{f}_{\text{fusion}} = [\mathbf{f}_{\text{FCNN}} \parallel \mathbf{p}_r]. \quad (22)$$

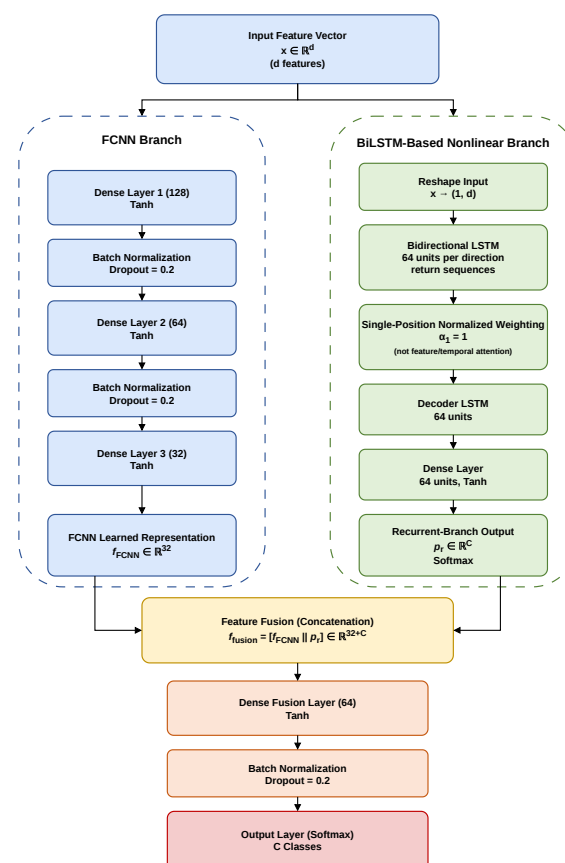


Figure 2. Architecture flow of the proposed parallel hybrid BiLSTM-FCNN (PH-BiLSTM-FCNN).

Since $\mathbf{f}_{\text{FCNN}} \in \mathbb{R}^{B \times 32}$ and $\mathbf{p}_r \in \mathbb{R}^{B \times C}$, the resulting fusion representation is

$$\mathbf{f}_{\text{fusion}} \in \mathbb{R}^{B \times (32+C)}. \quad (23)$$

The concatenated representation is subsequently processed through a 64-neuron non-linear fusion layer:

$$\mathbf{h}_{\text{fusion}} = \tanh(\mathbf{W}_{\text{fusion}} \mathbf{f}_{\text{fusion}} + \mathbf{b}_{\text{fusion}}), \quad (24)$$

where

$$\mathbf{h}_{\text{fusion}} \in \mathbb{R}^{B \times 64}. \quad (25)$$

Batch normalization and dropout with a rate of 0.2 are applied to this fused representation before final classification.

For a classification problem containing C traffic classes, the final predicted class distribution is obtained as

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{W}_o \mathbf{h}_{\text{fusion}} + \mathbf{b}_o), \quad (26)$$

where

$$\hat{\mathbf{y}} \in \mathbb{R}^{B \times C}. \quad (27)$$

The complete network is jointly optimized using categorical cross-entropy:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{B} \sum_{i=1}^B \sum_{c=1}^C y_{i,c} \log(\hat{y}_{i,c}), \quad (28)$$

where $y_{i,c}$ represents the ground-truth class indicator and $\hat{y}_{i,c}$ represents the corresponding predicted probability. The parameters of the FCNN branch, single-step BiLSTM-based nonlinear branch, fusion layer, and final classifier are optimized jointly through backpropagation using stochastic gradient descent.

Accordingly, the architectural contribution of PH-BiLSTM-FCNN lies in the parallel integration of two heterogeneous nonlinear learning pathways. The FCNN pathway learns hierarchical dense representations directly from the original tabular features, whereas the recurrent pathway provides an independently parameterized single-step nonlinear transformation of the complete traffic observation. Their integration allows complementary representations to contribute jointly to the final intrusion-classification decision.

In addition to predictive modeling, PH-BiLSTM-FCNN is integrated with the permissioned blockchain layer for auditable recording of model-related metadata and final evaluation information. The blockchain mechanism operates independently of the underlying learning architecture and does not modify the feature-learning process. Following model finalization, authorized model metadata, evaluation measurements, timestamps, and cryptographic digests are committed to the ledger to support traceability and integrity verification within the evaluated permissioned environment. Therefore, the blockchain component is interpreted as an audit and record-integrity mechanism rather than as part of the classification architecture. The complete architectural configuration of PH-BiLSTM-FCNN is summarized in Table 2, while its training and blockchain-assisted audit procedure is presented in Algorithm 1.

Table 2. Architecture of the proposed parallel hybrid BiLSTM-FCNN (PH-BiLSTM-FCNN).

Module	Layer	Configuration
FCNN Branch		
FCNN	Input	d predictor variables
	Dense	128 units, tanh
	Batch Normalization	Normalization of hidden activations
	Dropout	Rate = 0.2

Table 2. Cont.

Module	Layer	Configuration
	Dense	64 units, tanh
	Batch Normalization	Normalization of hidden activations
	Dropout	Rate = 0.2
	Dense	32 units, tanh
single-step BiLSTM-Based Nonlinear Branch		
BiLSTM Branch	Input Reshaping	$(1, d)$ single-position representation
	BiLSTM	64 units per direction, return sequences
	Normalized Weighting	Single position; $\alpha_1 = 1$
	Decoder LSTM	64 units
	Dense	64 units, tanh
	Branch Output	C units, softmax
Feature Fusion and Classification		
Fusion	Concatenation	$\mathbf{f}_{\text{FCNN}} \parallel \mathbf{p}_r \in \mathbb{R}^{32+C}$
	Dense	64 units, tanh
	Batch Normalization	Normalization of fused activations
	Dropout	Rate = 0.2
Output	Dense	C units, softmax
Training Configuration		
Training	Epochs	10
	Batch Size	128
	Optimizer	SGD
	Loss Function	Categorical cross-entropy

3.5.1. Proposed Q-Learning Optimized PH-BiLSTM-FCNN (Q-PH-BiLSTM-FCNN) Model

Although the proposed PH-BiLSTM-FCNN effectively combines complementary non-linear representations learned through parallel FCNN and single-step BiLSTM-based pathways, its performance remains sensitive to training hyperparameters such as learning rate and batch size. In complex intrusion detection scenarios characterized by heterogeneous traffic patterns and multi-class imbalance, fixed hyperparameter settings may lead to suboptimal convergence or poor generalization. Conventional hyperparameter-search strategies evaluate candidate configurations without explicitly conditioning subsequent decisions on the intermediate optimization behavior of the model. To address this limitation, Q-learning is integrated with PH-BiLSTM-FCNN to formulate hyperparameter selection as a sequential state-dependent optimization process.

Q-learning is employed as a model-free reinforcement-learning technique to guide sequential hyperparameter optimization. In the revised formulation, the PH-BiLSTM-FCNN training procedure is treated as an environment in which the agent observes an optimization state s_t , selects an action a_t , receives a reward r_t based on validation feedback, and subsequently transitions to a new state s_{t+1} . Each episode corresponds to a complete training cycle of PH-BiLSTM-FCNN under the selected hyperparameter configuration. Unlike the original action-only formulation $Q(a)$, the revised formulation explicitly models

the dependence between the current optimization state and the selected action through $Q(s, a)$. The optimization state at episode t is represented as

$$s_t = [P_t, \Delta P_t, U_t, G_t], \quad (29)$$

where P_t denotes the current validation performance, $\Delta P_t = P_t - P_{t-1}$ represents the variation in validation performance between consecutive episodes, U_t denotes predictive uncertainty on the validation partition, and $G_t = t/E$ represents the relative optimization progress, where E is the total number of Q-learning episodes. The resulting state therefore informs the agent about the current quality, direction, uncertainty, and progress of the optimization procedure.

Predictive uncertainty is quantified through the mean normalized entropy of the validation predictions:

$$U_t = \frac{1}{N_{\text{val}}} \sum_{i=1}^{N_{\text{val}}} \frac{-\sum_{c=1}^C p_{i,c} \log(p_{i,c})}{\log C}, \quad (30)$$

where $p_{i,c}$ denotes the predicted probability that validation sample i belongs to class c , C represents the number of traffic classes, and N_{val} is the number of validation samples. Normalization by $\log C$ constrains the uncertainty measure to approximately the interval $[0, 1]$.

Since tabular Q-learning requires a finite state representation, the continuous state variables are mapped into predefined discrete categories before Q-table indexing. Validation performance is discretized according to its observed level; ΔP_t indicates whether validation performance is improving, stable, or declining; predictive uncertainty is mapped to discrete uncertainty levels; and G_t represents the early, intermediate, or late stage of optimization. The resulting discrete state is used jointly with the selected action to index $Q(s_t, a_t)$.

The agent selects an action $a_t \in \mathcal{A}$, where every action represents one candidate combination of model learning rate and batch size. The action space is defined as

$$\mathcal{A} = \{(\eta_i, b_j)\}, \quad \eta \in \{10^{-4}, 10^{-3}, 10^{-2}\}, \quad b \in \{32, 64\}, \quad (31)$$

where η denotes the PH-BiLSTM-FCNN learning rate and b denotes the batch size [20]. At episode t , action selection is conditioned on the observed optimization state s_t . After executing action a_t , PH-BiLSTM-FCNN is trained using the corresponding learning-rate and batch-size configuration and is subsequently evaluated on the fixed validation partition. The obtained validation measurements are then used to construct the next state s_{t+1} . The resulting state transition is expressed as

$$s_t \xrightarrow{a_t} s_{t+1}. \quad (32)$$

The state-action value is updated using the standard Q-learning update rule:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_{a'} Q(s_{t+1}, a') - Q(s_t, a_t) \right], \quad (33)$$

where α denotes the Q-learning update rate, γ denotes the discount factor controlling the contribution of future rewards, and r_t represents the reward obtained after executing action a_t in state s_t .

The reward signal incorporates current validation performance, improvement relative to the previous episode, and computational cost:

$$r_t = \lambda_1 P_t + \lambda_2 \Delta P_t - \lambda_3 \tilde{T}_t, \quad (34)$$

where P_t represents the validation classification performance at episode t , ΔP_t rewards improvement relative to the preceding episode, $\tilde{T}_t$ denotes normalized model-training time, and λ_1 , λ_2 , and λ_3 control the relative contribution of predictive quality, performance improvement, and computational efficiency.

Formally, the state-dependent Q-function represents the expected cumulative discounted reward obtained by selecting action a while the optimization environment is in state s :

$$Q(s, a) = \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \mid s_t = s, a_t = a \right]. \quad (35)$$

Validation-set classification performance is used throughout Q-learning-based hyperparameter optimization. The held-out test set remains completely isolated throughout the optimization process and is accessed only once after the optimal hyperparameter configuration has been selected and the final Q-PH-BiLSTM-FCNN model has been fixed.

An ϵ -greedy strategy is adopted to balance exploration and exploitation. With probability ϵ , the agent randomly selects an available hyperparameter configuration to explore the action space. With probability $1 - \epsilon$, it selects the action associated with the largest state-dependent Q-value:

$$a_t = \arg \max_{a \in \mathcal{A}} Q(s_t, a). \quad (36)$$

Therefore, the selected hyperparameter configuration depends on the current optimization state rather than solely on a global action score.

To further enhance auditability and result integrity, Q-PH-BiLSTM-FCNN illustrated in Figure 3 is integrated with a blockchain-assisted logging mechanism and smart-contract layer. After each Q-learning episode, the trained model is evaluated on the fixed validation partition. The corresponding optimization state, selected action, validation performance, reward, hyperparameters, and execution metadata are provided to the smart contract. Upon successful authorization, these optimization records are committed to the permissioned blockchain ledger. The blockchain component therefore provides an auditable record of the sequential optimization process without directly participating in model learning.

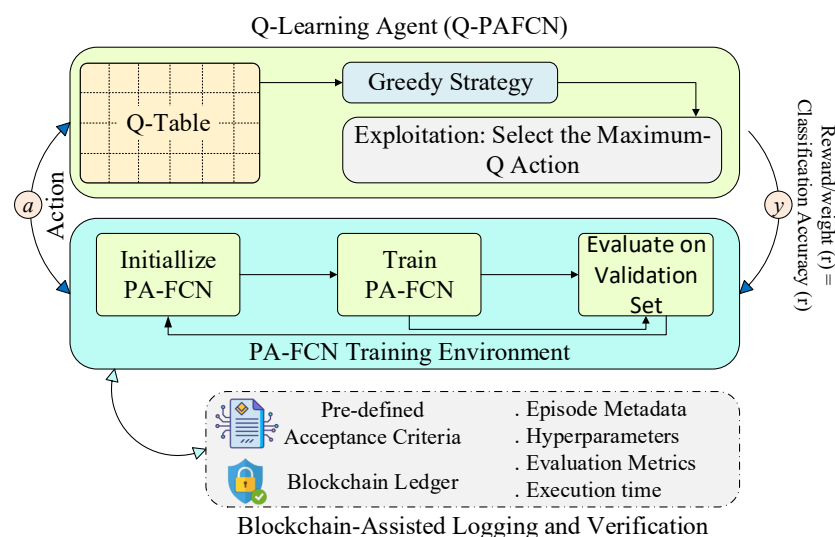


Figure 3. Working of proposed Q-PH-BiLSTM-FCNN model

During each episode, the PH-BiLSTM-FCNN model is trained for a fixed number of epochs using the selected hyperparameters and evaluated exclusively on the fixed validation partition. The resulting validation performance is used to construct the optimization state, compute the reward, update the Q-table, and guide subsequent hyperparameter selec-

tion. The held-out test partition is not accessed during any Q-learning episode and is evaluated only once after the optimal action and final model configuration have been fixed. The overall configuration of Q-PH-BiLSTM-FCNN is summarized in Table 3, while its complete optimization procedure is presented in Algorithm 2. The revised Q-PH-BiLSTM-FCNN therefore formulates Q-learning as a state-dependent sequential optimization procedure rather than an action-only hyperparameter-ranking mechanism. The selected learning-rate and batch-size configurations depend on the observed validation behavior, uncertainty, and optimization progress of the model. However, the current optimization is performed within an offline training and validation setting; therefore, the reported results should not be interpreted as evidence of real-time adaptation to continuously changing or previously unseen network traffic.

Table 3. Configuration of the state-dependent Q-PH-BiLSTM-FCNN model.

Hyperparameter	Setting
Q-Learning Control	
Episodes (E)	10
Epochs per Episode	10
Action Space ($\mathcal{A}$)	(η, b) pairs
Environment State	
Validation Performance	P_t
Performance Variation	ΔP_t
Predictive Uncertainty	U_t
Optimization Progress	$G_t = t/E$
Q-Learning Policy	
Q-Learning Update Rate (α)	0.1
Discount Factor (γ)	0.9
Exploration Rate (ϵ)	0.1
Optimized Parameters	
Candidate Learning Rates (η)	$\{10^{-4}, 10^{-3}, 10^{-2}\}$
Candidate Batch Sizes (b)	$\{32, 64\}$

3.5.2. Proposed Entropy-Based Active Learning on PH-BiLSTM-FCNN (EAL-PH-BiLSTM-FCNN)

While the PH-BiLSTM-FCNN demonstrates strong classification capability by combining sequence-aware attention with dense feature extraction, its effectiveness remains strongly dependent on the availability of labeled data. In real-world IDSs, labeling large-scale network traffic is costly, time-consuming, and often impractical. To alleviate this limitation, we integrate EAL with PH-BiLSTM-FCNN, resulting in the EAL-PH-BiLSTM-FCNN framework, which selectively queries the most informative samples for annotation while minimizing labeling effort.

In the proposed EAL-PH-BiLSTM-FCNN, model uncertainty for each unlabeled instance is quantified using Shannon entropy [21], computed over the predicted class probability distribution, as Figure 4 shows

$$H(x_i) = - \sum_{c=1}^C p_{i,c} \log p_{i,c}, \quad (37)$$

where $p_{i,c}$ denotes the predicted probability of sample x_i belonging to class c , and C is the total number of classes. Samples exhibiting higher entropy values indicate greater

predictive uncertainty and are therefore prioritized for labeling.

Before initiating active learning, the development data are separated into a training partition and a fixed validation partition, while the held-out test partition is reserved exclusively for the final evaluation. The active-learning pool is constructed only from the training partition, which is stratified into an initial labeled subset $X_L^{(0)}$ and an unlabeled pool $X_U^{(0)}$. Consequently, neither validation nor test observations participate in the sample-acquisition process.

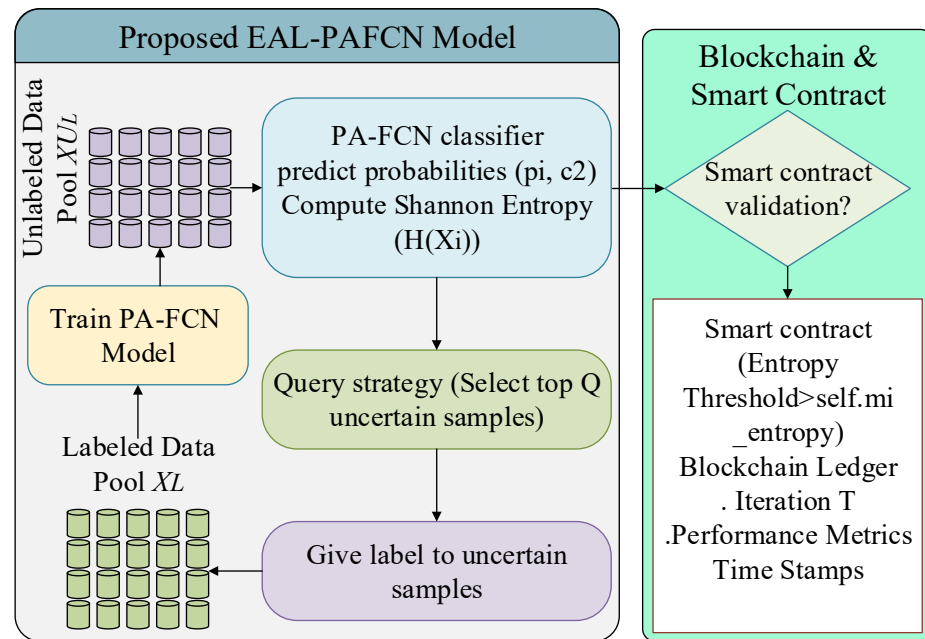


Figure 4. Working of proposed EAL-PH-BiLSTM-FCNN model

At each active-learning iteration t , PH-BiLSTM-FCNN is trained solely using the currently labeled set $X_L^{(t)}$. The trained model is then evaluated on the fixed validation partition to monitor iteration-level learning behavior, whereas the test partition remains completely inaccessible during active sample acquisition. Predictive probabilities are subsequently obtained for all samples in $X_U^{(t)}$, and their entropy values are computed to identify uncertain instances.

In accordance with the implemented acquisition protocol, the maximum query size is fixed relative to the initial labeled pool rather than recalculated from the progressively enlarged labeled set. Specifically,

$$Q = 0.1 \times |X_L^{(0)}|. \quad (38)$$

A smart-contract entropy threshold $\tau = 1.0$ is first applied to identify the set of authorized uncertain samples:

$$\mathcal{A}_t = \{x \in X_U^{(t)} \mid H(x) \geq \tau\}. \quad (39)$$

The authorized samples are then ranked according to their entropy values, and at most the top- Q highest-entropy observations are selected to form the query set $\mathcal{Q}_t$:

$$\mathcal{Q}_t = \text{TopQ}_{x \in \mathcal{A}_t} H(x), \quad |\mathcal{Q}_t| \leq Q. \quad (40)$$

The inequality $|\mathcal{Q}_t| \leq Q$ accounts for iterations in which fewer than Q unlabeled observations satisfy the predefined entropy threshold.

Once their labels are acquired, the selected instances are transferred from the unlabeled pool to the labeled pool according to

$$X_L^{(t+1)} \leftarrow X_L^{(t)} \cup \mathcal{Q}_t, \quad X_U^{(t+1)} \leftarrow X_U^{(t)} \setminus \mathcal{Q}_t. \quad (41)$$

This iterative enrichment strategy progressively refines the model's decision boundaries by concentrating supervision on uncertain and informative traffic observations. Model optimization at each iteration follows the standard categorical cross-entropy objective:

$$\mathcal{L} = - \sum_{i=1}^N \sum_{c=1}^C y_{i,c} \log \hat{y}_{i,c}, \quad (42)$$

where $y_{i,c}$ and $\hat{y}_{i,c}$ denote the true and predicted class probabilities, respectively. In our experiments, the active-learning process is executed for $T = 5$ iterations, using $E = 10$ training epochs per iteration and a batch size of $B = 128$.

The initial training pool is divided into two equally sized subsets, with 50% forming the initial labeled pool $X_L^{(0)}$ and the remaining 50% forming the unlabeled pool $X_U^{(0)}$. Since the maximum query size in each iteration is fixed at 10% of the initial labeled-pool size, the maximum attainable labeled proportion after five acquisition rounds is 75% of the training partition. However, the actual labeling ratio may be lower because only observations satisfying the entropy threshold $\tau = 1.0$ are eligible for acquisition.

To ensure transparency, integrity, and traceability of the active-learning process, EAL-PH-BiLSTM-FCNN is further augmented with a lightweight blockchain and a smart contract layer. After each active-learning iteration, only validation-set performance metrics and query statistics are recorded on the blockchain. No test-set performance is computed or used during the iterative acquisition process. The smart contract authorizes candidate samples according to the predefined entropy threshold, after which the highest-entropy authorized observations are selected up to the fixed query budget.

After completion of all T active-learning iterations, the final labeling budget and enriched labeled pool are fixed. A final EAL-PH-BiLSTM-FCNN model is then trained using this fixed labeled set. Only at this stage is the previously untouched test partition accessed, and the finalized model is evaluated once to obtain the reported final test-set performance. These final test metrics are recorded separately on the blockchain from the iteration-level validation records. This design ensures that the test partition does not influence sample acquisition, labeling-budget determination, or intermediate model assessment, thereby preserving an independent final evaluation. The complete blockchain-enabled EAL procedure is summarized in Algorithm 3. Through iterative uncertainty-driven querying, validation-based monitoring, and immutable logging, EAL-PH-BiLSTM-FCNN aims to reduce labeling requirements while preserving an independent assessment of final classification performance.

3.5.3. Threat Model and Security Scope

The blockchain layer is introduced as an audit and integrity mechanism for security-relevant records generated during the proposed IDS workflow rather than as a mechanism for securing the underlying classifier itself. The protected assets include model-execution records, active-learning query information, validation and final evaluation metadata, timestamps, transaction identifiers, and cryptographic digests associated with the recorded operations. These records are committed to a permissioned ledger using hash-linked blocks, authenticated transactions, and PBFT-style validator voting. Consequently, modification of an already committed record changes its cryptographic digest and breaks the

corresponding blockchain linkage, enabling integrity violations to be detected during ledger verification.

The considered threat model includes an unauthorized entity attempting to submit or modify audit records and a faulty or potentially malicious validator attempting to participate inconsistently in ledger validation. The evaluated permissioned configuration consists of four authenticated validators with a quorum of three and is configured to tolerate at most one faulty validator. However, this assumption represents the intended fault model of the local PBFT-style prototype and should not be interpreted as experimental evidence of resilience against arbitrary Byzantine attacks or validator collusion.

Several components remain outside the protection boundary of the blockchain. In particular, the raw network traffic, correctness of the original labels, integrity of the data-collection devices, correctness of the trained PH-BiLSTM-FCNN predictions, and security of the endpoints generating blockchain transactions are assumed to be trusted before their corresponding metadata are submitted to the ledger. Similarly, the blockchain does not by itself prevent adversarial examples, data poisoning, model extraction, compromised sensing devices, or incorrect information supplied by an otherwise authorized writer. Raw traffic data and model parameters are maintained off-chain, while only the associated audit information and integrity-related metadata are recorded on-chain. Therefore, the security objective of the present blockchain component is limited to providing authenticated, traceable, and integrity-verifiable audit records within the evaluated permissioned environment. Stronger claims regarding resilience to malicious-validator collusion, forged or replayed transactions, conflicting updates, multiple validator failures, and network partitions require dedicated adversarial and fault-injection experiments. These scenarios are consequently identified as part of the future security evaluation of the framework.

4. Results and Simulation

This section summarizes the experimental estimation and simulation findings of the proposed models, demonstrating their effectiveness for IoT intrusion detection under a controlled and reproducible evaluation environment. The experiments provide a fair comparison among the proposed models and different categories of baselines, including conventional machine learning models, deep learning architectures, and existing approaches. In particular, the proposed models demonstrate improved predictive performance compared with the adopted deep learning baselines, including FCNN, LSTM, GRU, and Bi-LSTM, as well as the Logistic Regression (LR) baseline. However, these results are interpreted within the controlled balanced benchmark and are not presented as evidence of superiority over all conventional tabular classifiers. The contribution of the proposed framework is instead attributed to the integration of nonlinear representation learning, validation-guided hyperparameter optimization, and label-efficient learning mechanisms for improving intrusion detection performance.

Experiments are all conducted in a controlled and reproducible environment to provide fair performance comparison with baseline, existing, and proposed approaches. The simulations are run on the platform of Google Colab, using the Python 3.12 programming language, and running on a 64-bit operating system with an $\times 64$ -based processor. The hardware setup is based on an Intel(R) Core(TM) i5-7300U processor with an operating frequency of 2.60 GHz, 16 GB RAM, and a 238 GB solid-state drive. Such arrangements offers adequate computing capabilities to train DL models and estimate realistic Execution Times (ET). All models are applied and tested under the same environmental conditions to provide consistency and reliability of the reported results.

Table 4 and Figure 5 report the comparative performance of existing, baseline, and proposed models in terms of accuracy, precision, recall, F1-score, Matthews Correlation Coef-

ficient (MCC), and Cohen’s Kappa (CK), providing a comprehensive evaluation of both predictive correctness and classification reliability. LR [22] has an accuracy of 0.8063 at the expense of 0.7566 and 0.7829 in terms of MCC and CK, respectively, and shows stable linear separability but low ability to learn more complex nonlinear intrusion patterns, thus lowering performance compared to deep models. The LSTM model achieves an accuracy of 0.7603 and a MCC of 0.7269, indicating its capability to capture temporal dependencies, but its performance is diminished by over-parameterization and sensitivity to long training periods with high-dimensional tabular traffic data. The GRU model further shows decreased accuracy of 0.7033 with MCC and CK values of 0.6617 and 0.6292, reflecting faster convergence than LSTM but insufficient representational power to discriminate subtle multi-class attack behaviors. The TabTransformer (TT) model achieves an accuracy of 0.5992, with MCC and CK values of 0.5583 and 0.4992, respectively. Although TT employs self-attention mechanisms to learn contextual relationships among tabular features, its relatively low F1-score of 0.4660 indicates difficulty in achieving balanced classification across different attack classes, which may be attributed to limited training data, class overlap, and the complexity of high-dimensional network traffic features. single-step BiLSTM nonlinear encoder is comparatively more successful with an accuracy of 0.8064 and MCC of 0.7625 because of the use of attention-based feature relevance learning, but the high cost of computation and the inability to discriminate non-linearly limit overall improvements. FCNN accuracy of 0.6299 and MCC and CK rates of 0.5859 and 0.5376 are lower because the features are independent of each other, with no mechanisms to capture contextual dependencies, resulting in suboptimal intrusion detection performance.

Table 4. Comparison of traditional, deep learning, and proposed models.

Metric	LR	LSTM	GRU	TT	Single-Step BiLSTM Nonlinear Encoder	FCNN	PH-BiLSTM- FCNN	Q-PH-BiLSTM- FCNN	EAL-PH- BiLSTM-FCNN
Accuracy	0.8063	0.7603	0.7033	0.5992	0.8064	0.6299	0.8451	0.9154	0.8616
Precision	0.8210	0.8229	0.8021	0.5935	0.8219	0.7631	0.8462	0.9208	0.8846
Recall	0.8263	0.7603	0.7033	0.5992	0.8063	0.6299	0.8451	0.9154	0.8616
F1-Score	0.8037	0.7303	0.6487	0.4660	0.8040	0.5389	0.8441	0.9144	0.8584
ROC-AUC	0.9210	0.9371	0.9075	0.9325	0.9590	0.9099	0.9654	0.9905	0.9765
PR-AUC	0.8121	0.7450	0.6850	0.7268	0.8063	0.6330	0.8542	0.9619	0.8813
LL	0.4348	0.5530	0.8007	0.5561	0.3777	0.5483	0.2485	0.1489	0.2977
MCC	0.7566	0.7269	0.6617	0.5583	0.7625	0.5859	0.8073	0.8962	0.9120
CK	0.7829	0.7004	0.6292	0.4992	0.7580	0.5376	0.8063	0.8942	0.8340
HL	0.1737	0.2397	0.2967	0.4008	0.1936	0.3701	0.1549	0.0846	0.1384
ET (s)	38.22	123.38	165.39	384.47	728.0183	240.97	799.26	1515.07	324.28

In contrast, the proposed PH-BiLSTM-FCNN significantly improves performance with an accuracy of 0.8451, precision of 0.8462, recall of 0.8451, F1-score of 0.8441, MCC of 0.8073, and CK of 0.8063, demonstrating that parallel fusion of FCNN-based nonlinear abstraction and single-step BiLSTM nonlinear encoder-based nonlinear feature transformation effectively enhances both predictive accuracy and class agreement. The Q-PH-BiLSTM-FCNN further achieves the best overall performance, attaining an accuracy of 0.9154, F1-score of 0.9144, MCC of 0.8962, and CK of 0.8942, where the primary advantage lies in RL-driven adaptive hyperparameter optimization that stabilizes convergence and maximizes discriminative learning. The EAL-PH-BiLSTM-FCNN model achieves a value of 0.8616 with MCC and CK values of 0.9120 and 0.8340 respectively, indicating that EAL is effective in boosting the reliability of the model by focusing on informative samples but reducing

labeling work, but it still has lower performance than Q-PH-BiLSTM-FCNN because there is limited labeling data in earlier learning cycles.

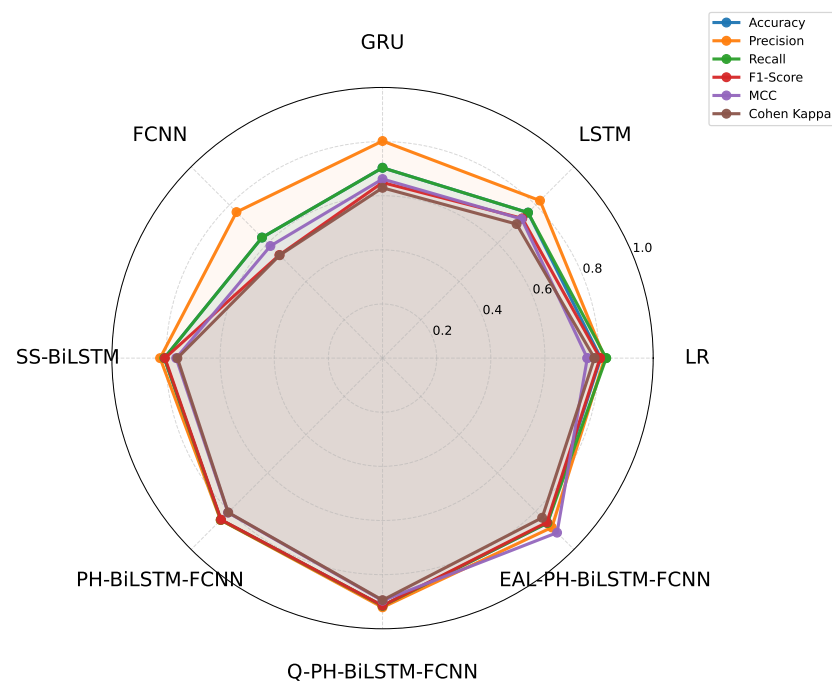


Figure 5. Performance comparison of existing, baseline, and proposed models.

Figure 6 presents a comparative analysis of the models in terms of Hamming Loss (HL), Log Loss (LL), and ET, offering insights into both error behavior and computational efficiency. Of the currently available models, LR has the smallest ET of 38.22 s with a HL of 0.1737 and a LL of 0.4348, indicating its computational simplicity and low learning ability. The ETs of LSTM and GRU are significantly larger, 123.38s and 165.39s, respectively, and HL and LL are greater, 0.2397 and 0.2967 respectively, showing inefficiencies in high-dimensional tabular traffic data modeling. The TT model records an ET of 384.47 s, with a HL of 0.4008 and a LL of 0.5561, indicating considerable computational requirements and a relatively high classification error rate despite its attention-based capability to capture contextual relationships among tabular features. single-step BiLSTM nonlinear encoder shows a sharp rise in ET to 728.0183 s with moderate HL of 0.1936 and LL of 0.3777, highlighting the computational overhead introduced by attention mechanisms. FCNN records an ET of 240.97 s but suffers from the highest HL of 0.3701 and LL of 0.5483, confirming its weaker classification reliability. Conversely, the proposed PH-BiLSTM-FCNN minimizes classification errors with lower HL of 0.1549 and LL of 0.2485, but with a higher ET of 799.26 s because of the parallel architecture. The Q-PH-BiLSTM-FCNN has the smallest HL of 0.0846 and the smallest LL of 0.1489, which proves to be confident and accurate in predictions; but at the expense of the largest ET of 1515.07 s due to the hyperparameter optimization by RL. The EAL-PH-BiLSTM-FCNN is a good compromise, as it reduces HL to 0.1384 and LL to 0.2977 without dramatically decreasing the comparatively moderate ET of 324.28 s, which proves that EAL enhances efficiency by prioritizing training on the most informative samples.

Figure 7 illustrates the Receiver Operating Characteristic–Area Under the Curve (ROC–AUC) performance comparison of existing, baseline, and proposed models, highlighting their capability to discriminate between normal and attack traffic across varying false positive rates. Among the existing approaches, LR achieves a ROC–AUC of 0.9210, indicating reasonable separability but limited robustness in complex intrusion scenarios.

The LSTM model slightly improves discrimination with a ROC–AUC of 0.9371 by capturing sequential dependencies, whereas the GRU records a lower ROC–AUC of 0.9075, reflecting reduced sensitivity to subtle class boundaries. The TT model achieves a ROC–AUC of 0.9325, demonstrating a reasonable ability to distinguish between normal and attack traffic through its attention-based learning of contextual relationships among tabular features, although its discriminative performance remains lower than that of the single-step BiLSTM nonlinear encoder. The attention-driven feature weighting of the single-step BiLSTM nonlinear encoder enhances its ability, boosting its ROC–AUC score to 0.9590, whereas the FCNN’s performance is 0.9099, indicating its comparatively inferior discriminative capability in the absence of contextual relationships. In contrast, the proposed PH-BiLSTM-FCNN further enhances detection performance with a ROC–AUC of 0.9654 by jointly exploiting nonlinear feature learning and attention-based context modeling. RL-driven hyperparameter optimization results in the highest ROC–AUC of 0.9905, which shows near-optimal class separability. The EAL-PH-BiLSTM-FCNN achieves a ROC–AUC of 0.9765, indicating that uncertainty-aware sample selection significantly enhances discrimination performance with only a small number of labeled instances.

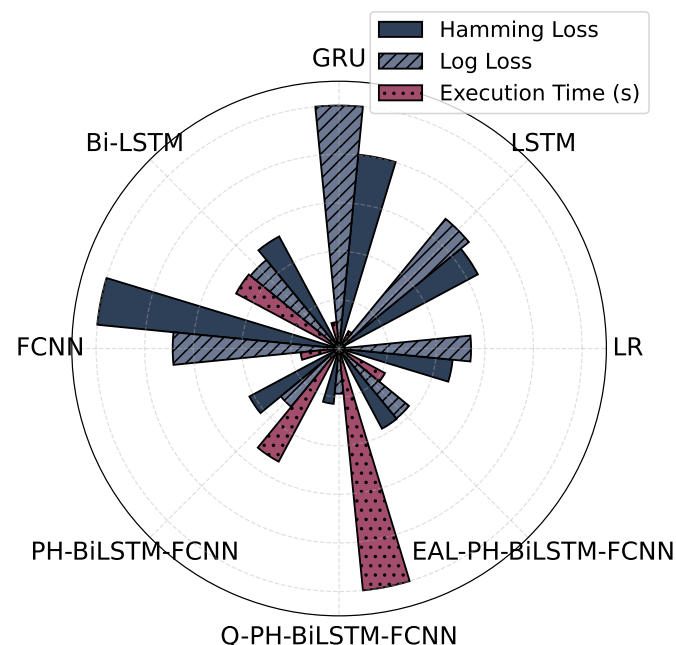


Figure 6. Comparison of Hamming Loss, Log Loss, and Execution Time for existing, baseline, and proposed models.

Figure 8 presents the Precision–Recall Area Under the Curve (PR–AUC) comparison of existing, baseline, and proposed models, emphasizing their effectiveness in handling class imbalance by evaluating the trade-off between precision and recall. We compare the existing models and we find that LR has a moderate reliability in detecting attack classes with PR–AUC of 0.8121, and it is less robust when the data distribution is skewed. The LSTM and GRU models achieve PR AUC of 0.7450 and 0.6850, respectively, which indicates a moderate decrease in precision at high values of recall, caused by overfitting and poor discrimination of minority attack samples. The TT model achieves a PR–AUC of 0.7268, indicating moderate performance in identifying minority attack classes. Although its self-attention mechanism captures contextual relationships among tabular features, its lower precision–recall balance reflects difficulty in consistently distinguishing overlapping and underrepresented attack samples. Despite the benefits of attention-based feature relevance learning, the single-step BiLSTM nonlinear encoder has high model complexity

and achieves a PR-AUC of 0.8063. The FCNN achieves the lowest PR-AUC score, 0.6330, to validate its limited performance in imbalanced intrusion detection scenarios that heavily rely on contextual data. In contrast, the proposed PH-BiLSTM-FCNN significantly improves performance with a PR-AUC of 0.8542 by jointly leveraging nonlinear abstraction and dense representation layers. The Q-PH-BiLSTM-FCNN achieves the highest PR-AUC of 0.9619, demonstrating superior precision–recall balance and strong minority class detection capability driven by RL-optimized training dynamics. The EAL-PH-BiLSTM-FCNN attains a PR-AUC of 0.8813, indicating that EAL effectively enhances precision on uncertain samples while reducing labeling overhead, albeit with slightly lower performance than fully optimized Q-PH-BiLSTM-FCNN.

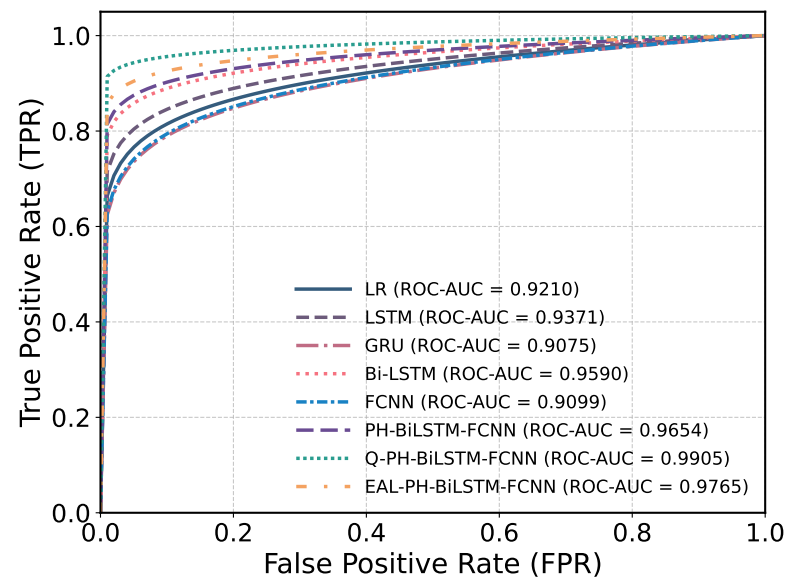


Figure 7. ROC-AUC comparison of existing, baseline, and proposed models.

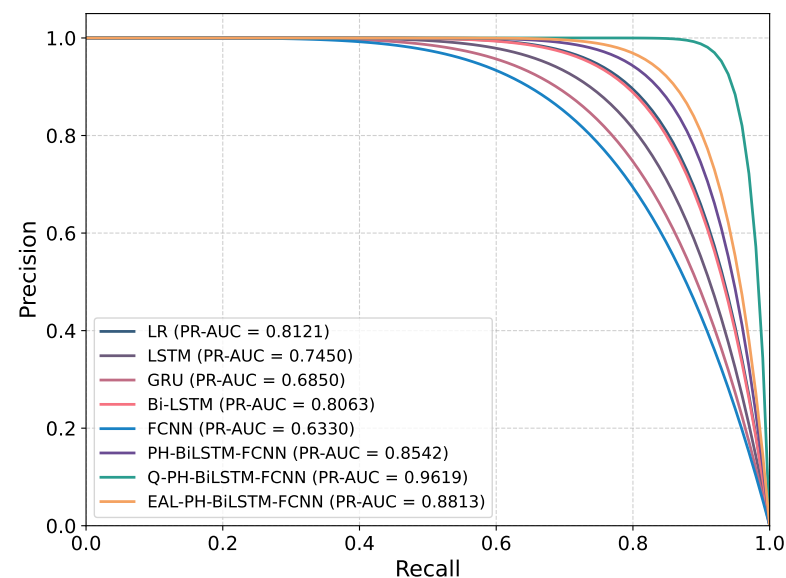


Figure 8. PR-AUC comparison of existing, baseline, and proposed models.

4.1. Macro-Averaged and Class-Wise Performance Analysis

Although the aggregate comparison in Table 4 provides an overall assessment of the evaluated models, weighted measures may obscure differences in class-specific detection performance. In particular, support-weighted recall is mathematically equivalent to

accuracy in a single-label multiclass classification problem, which explains the identical accuracy and recall values observed for several models. Therefore, additional macro-averaged and class-wise evaluations are performed for PH-BiLSTM-FCNN, Q-PH-BiLSTM-FCNN, and EAL-PH-BiLSTM-FCNN. The original aggregate accuracies of PH-BiLSTM-FCNN, Q-PH-BiLSTM-FCNN, and EAL-PH-BiLSTM-FCNN remain 0.8451, 0.9154, and 0.8616, respectively, as reported in Table 4. The following analysis presents the additional class-sensitive measures derived from the corresponding confusion matrices.

Table 5 shows that the macro-averaged metrics provide a more class-sensitive assessment than the aggregate weighted measures. Minority mean recall is calculated as the unweighted mean recall of Classes 0–3, while Class 4 is treated as the majority class. The corresponding class-wise precision, recall, F1-score, and support values are reported in Table 6.

Table 5. Macro-averaged and minority-class performance of the proposed models.

Model	Macro Precision	Macro Recall	Macro F1-Score	Minority Mean Recall
PH-BiLSTM-FCNN	0.8195	0.7795	0.7424	0.7244
Q-PH-BiLSTM-FCNN	0.8712	0.8299	0.8230	0.7874
EAL-PH-BiLSTM-FCNN	0.9036	0.8872	0.8855	0.8590

Table 6. Class-wise precision, recall, F1-score, and support of PH-BiLSTM-FCNN, Q-PH-BiLSTM-FCNN, and EAL-PH-BiLSTM-FCNN.

Model	Class	Precision	Recall	F1-Score	Support	Class Status
PH-BiLSTM-FCNN	Class 0	0.8410	0.9999	0.9136	27,787	Minority
	Class 1	1.0000	0.8118	0.8961	27,937	Minority
	Class 2	0.7338	0.1346	0.2274	27,837	Minority
	Class 3	0.5230	0.9513	0.6749	27,761	Minority
	Class 4	1.0000	1.0000	1.0000	27,751	Majority
Q-PH-BiLSTM-FCNN	Class 0	0.7076	1.0000	0.8287	27,787	Minority
	Class 1	1.0000	0.5889	0.7413	27,937	Minority
	Class 2	0.9440	0.5962	0.7308	27,837	Minority
	Class 3	0.7043	0.9645	0.8141	27,761	Minority
	Class 4	1.0000	1.0000	1.0000	27,751	Majority
EAL-PH-BiLSTM-FCNN	Class 0	1.0000	0.7009	0.8241	27,815	Minority
	Class 1	0.7698	0.9997	0.8699	27,814	Minority
	Class 2	0.8238	0.9353	0.8760	27,815	Minority
	Class 3	0.9248	0.8002	0.8580	27,815	Minority
	Class 4	0.9998	0.9996	0.9997	27,814	Majority

For PH-BiLSTM-FCNN, the highest recall is obtained for Classes 0 and 4, whereas Class 2 exhibited substantially lower recall, indicating that many Class 2 observations are assigned to Class 3. Q-PH-BiLSTM-FCNN improved the recall of Class 2 but reduced the recall of Class 1. EAL-PH-BiLSTM-FCNN provided a more balanced class-wise recall distribution, with recalls of 0.7009, 0.9997, 0.9353, 0.8002, and 0.9996 for Classes 0–4, respectively.

The raw confusion matrices in Table 7 indicate that PH-BiLSTM-FCNN primarily confused Class 2 with Class 3, while Q-PH-BiLSTM-FCNN exhibited misclassification between Classes 0 and 1 and between Classes 2 and 3. EAL-PH-BiLSTM-FCNN reduced the Class 2–Class 3 confusion but showed additional confusion between Classes 0 and 1. To facilitate comparison independent of class support, the corresponding row-normalized confusion matrices are presented in Table 8.

Table 7. Raw confusion matrices of PH-BiLSTM-FCNN, Q-PH-BiLSTM-FCNN, and EAL-PH-BiLSTM-FCNN. Rows denote actual classes, whereas columns denote predicted classes.

(a) PH-BiLSTM-FCNN					
Actual	P0	P1	P2	P3	P4
Class 0	27,783	0	4	0	0
Class 1	5254	22,679	4	0	0
Class 2	0	0	3746	24,091	0
Class 3	0	0	1351	26,410	0
Class 4	0	0	0	0	27,751
(b) Q-PH-BiLSTM-FCNN					
Actual	P0	P1	P2	P3	P4
Class 0	27,787	0	0	0	0
Class 1	11,485	16,452	0	0	0
Class 2	0	0	16,596	11,241	0
Class 3	0	0	985	26,776	0
Class 4	0	0	0	0	27,751
(c) EAL-PH-BiLSTM-FCNN					
Actual	P0	P1	P2	P3	P4
Class 0	19,495	8314	2	0	4
Class 1	0	27,807	4	2	1
Class 2	0	0	26,016	1799	0
Class 3	0	0	5558	22,257	0
Class 4	0	0	0	10	27,804

Table 8. Row-normalized confusion matrices of PH-BiLSTM-FCNN, Q-PH-BiLSTM-FCNN, and EAL-PH-BiLSTM-FCNN.

(a) PH-BiLSTM-FCNN					
Actual	P0	P1	P2	P3	P4
Class 0	0.9999	0.0000	0.0001	0.0000	0.0000
Class 1	0.1881	0.8118	0.0001	0.0000	0.0000
Class 2	0.0000	0.0000	0.1346	0.8654	0.0000
Class 3	0.0000	0.0000	0.0487	0.9513	0.0000
Class 4	0.0000	0.0000	0.0000	0.0000	1.0000
(b) Q-PH-BiLSTM-FCNN					
Actual	P0	P1	P2	P3	P4
Class 0	1.0000	0.0000	0.0000	0.0000	0.0000
Class 1	0.4111	0.5889	0.0000	0.0000	0.0000
Class 2	0.0000	0.0000	0.5962	0.4038	0.0000
Class 3	0.0000	0.0000	0.0355	0.9645	0.0000
Class 4	0.0000	0.0000	0.0000	0.0000	1.0000
(c) EAL-PH-BiLSTM-FCNN					
Actual	P0	P1	P2	P3	P4
Class 0	0.7009	0.2989	0.0001	0.0000	0.0001
Class 1	0.0000	0.9997	0.0001	0.0001	0.0000
Class 2	0.0000	0.0000	0.9353	0.0647	0.0000
Class 3	0.0000	0.0000	0.1998	0.8002	0.0000
Class 4	0.0000	0.0000	0.0000	0.0004	0.9996

4.2. Computational Performance and Overhead Analysis of the Permissioned Blockchain

To evaluate the practical computational impact of the blockchain layer, a separate benchmark is conducted for the custom local permissioned blockchain integrated with the proposed framework. The blockchain employed four validators and a PBFT-style validator-voting mechanism, with a fault-tolerance level of one and a minimum quorum of three validators. A total of 100 authorized transactions are submitted to measure transaction latency, throughput, storage consumption, CPU time, memory utilization, and the additional overhead introduced during model training and inference. Table 9 summarizes the resulting blockchain performance measurements.

Table 9. Computational performance and operational overhead of the permissioned blockchain component.

Blockchain Metric	Measured Value
Blockchain platform	Custom local permissioned blockchain
Consensus mechanism	PBFT-style validator voting
Validators/quorum	4/3
Fault tolerance	1
Benchmark transactions	100
Commit success rate	100.00%
Average transaction latency (ms)	4.7475
Median transaction latency (ms)	4.6971
95th-percentile latency (ms)	5.4065
Transaction throughput (tx/s)	198.41
Initial blockchain storage (KB)	0.3086
Final blockchain storage (KB)	72.5586
Storage growth (KB)	72.2500
Storage growth per transaction (bytes)	739.84
Average CPU time per transaction (ms)	4.7402
Peak benchmark memory (KB)	113.0908
Blockchain time during training (s)	0.019818
Blockchain share of training time	0.002864%
Observed training-time difference (s)	−13.4285
Observed training-time difference	−1.904%
Direct inference audit overhead (ms)	19.4526
Direct audit overhead relative to inference	0.3417%
Observed end-to-end inference difference (ms)	214.7835
Operational blocks stored	22
Blockchain integrity verification	Valid

The blockchain successfully committed all 100 benchmark transactions, resulting in a commit success rate of 100%. The average transaction latency is 4.7475 ms, while the median and 95th-percentile latencies are 4.6971 ms and 5.4065 ms, respectively. These closely grouped latency values indicate stable transaction processing under the evaluated local configuration. The blockchain achieved a throughput of 198.41 transactions per second, while the average CPU time required for each transaction is 4.7402 ms. The peak memory consumption during the benchmark is 113.0908 KB, indicating that the local validation and hashing operations introduced limited memory demand.

The blockchain storage increased from 0.3086 KB to 72.5586 KB after processing the benchmark transactions, corresponding to an average growth of 739.84 bytes per committed transaction. This increase reflects the storage of transaction payloads, SHA3-256 hashes, authorization signatures, block metadata, and validator-voting records. Although blockchain

storage grows with the number of recorded transactions, the measured increase remains moderate for the evaluated training and audit workload.

During model training, blockchain-assisted epoch logging consumed only 0.019818 s, representing approximately 0.002864% of the total training time. The blockchain-enabled run is 13.4285 s faster than the independently executed baseline run. However, this negative difference should not be interpreted as a computational acceleration caused by blockchain integration, because training time can vary due to hardware scheduling, memory allocation, TensorFlow initialization, and other runtime effects. Therefore, the directly measured blockchain callback time provides a more reliable estimate of the blockchain training overhead.

For inference auditing, generating a prediction digest and committing its corresponding transaction introduced an average direct overhead of 19.4526 ms, equivalent to approximately 0.3417% of the model inference time. The larger observed end-to-end difference of 214.7835 ms includes normal variation in model inference execution in addition to blockchain processing and should therefore be interpreted cautiously. The operational blockchain stored 22 blocks, and the final integrity verification returned a valid status, confirming that block hashes, transaction signatures, previous-block references, and consensus requirements remained consistent throughout training, evaluation, and inference auditing. Overall, the results demonstrate that the permissioned blockchain provides tamper-resistant model logging and verifiable execution records with relatively small directly measured computational overhead. Nevertheless, these findings represent a local PBFT-style simulation rather than a physically distributed blockchain deployment; therefore, network communication latency, node failures, and geographically distributed consensus costs are not reflected in the reported measurements.

4.3. Independent External-Dataset Evaluation on the CICIOT2023 Dataset

To examine whether the proposed architectures remain effective beyond the primary WUSTL-IIoT-2021 benchmark, an independent evaluation is conducted on the CICIOT2023 dataset [23]. This experiment should be interpreted as an external-dataset, within-dataset evaluation rather than a direct cross-dataset transfer experiment. Specifically, the proposed models are retrained from scratch using the CICIOT2023 training partition and are subsequently evaluated on a held-out CICIOT2023 evaluation partition. No trained model weights, fitted preprocessing parameters, or learned representations from WUSTL-IIoT-2021 are transferred to CICIOT2023. CICIOT2023 dataset contains more than one million IoT traffic records represented by 47 flow-based, packet-level, and temporal features. The dataset covers diverse benign and malicious activities, including DoS, DDoS, brute-force, spoofing, reconnaissance, web-based attacks, and Mirai malware variants. During preprocessing, categorical attack labels are converted into numerical class identifiers. A stratified subset of 150,000 records is then selected to preserve the original class distribution, followed by feature scaling and separation of the predictors and target labels. The same autoencoder-based balancing mechanism employed in the proposed framework is applied to the training data, while the evaluation data retains original records to provide an unbiased assessment of generalization performance.

Table 10 presents the performance of single-step BiLSTM nonlinear encoder, FCNN, PH-BiLSTM-FCNN, Q-PH-BiLSTM-FCNN, and EAL-PH-BiLSTM-FCNN on the CICIOT2023 dataset. The standalone single-step BiLSTM nonlinear encoder and FCNN models achieve accuracies of 0.6384 and 0.6299, respectively, with F1-scores of 0.5998 and 0.6031. Combining the two architectures in PH-BiLSTM-FCNN improves the accuracy to 0.7114, precision to 0.7266, F1-score to 0.6914, ROC-AUC to 0.9837, and PR-AUC to 0.7611. PH-BiLSTM-FCNN

also reduces HL and LL to 0.2886 and 0.8160, respectively, demonstrating the advantage of feature fusion over the individual models.

Table 10. Generalizability performance comparison on the CICIOT2023 dataset.

Metric	Single-Step BiLSTM	FCNN	PH-BiLSTM-FCNN	Q-PH-BiLSTM-FCNN	EAL-PH-BiLSTM-FCNN
Accuracy	0.6384	0.6299	0.7114	0.9544	0.8927
Precision	0.6059	0.6063	0.7266	0.9614	0.9038
Recall	0.6384	0.6299	0.7114	0.9544	0.8927
F1-score	0.5998	0.6031	0.6914	0.9536	0.8895
ROC-AUC	0.9787	0.9727	0.9837	0.9993	0.9981
PR-AUC	0.6983	0.7040	0.7611	0.9792	0.9483
CK	0.6275	0.6186	0.7026	0.9530	0.8895
MCC	0.6406	0.6515	0.7073	0.9533	0.8900
HL	0.3616	0.3701	0.2886	0.0456	0.1073
LL	1.0425	1.0012	0.8160	0.1043	0.2666

The Q-PH-BiLSTM-FCNN model provides the strongest generalization performance across all reported measures. It achieves an accuracy of 0.9544, precision of 0.9614, recall of 0.9544, and F1-score of 0.9536. Its ROC-AUC and PR-AUC reach 0.9993 and 0.9792, respectively, while the CK and MCC values increase to 0.9530 and 0.9533. Furthermore, Q-PH-BiLSTM-FCNN records the lowest HL of 0.0456 and LL of 0.1043, indicating both accurate class prediction and reliable probability estimation. EAL-PH-BiLSTM-FCNN also demonstrates strong transferable performance, obtaining an accuracy of 0.8927, F1-score of 0.8895, ROC-AUC of 0.9981, and PR-AUC of 0.9483. Its HL and LL are limited to 0.1073 and 0.2666, respectively. These results show that the improvements obtained by the proposed PH-BiLSTM-FCNN variants are maintained on a large and heterogeneous external IoT dataset, supporting the generalizability and robustness of the framework across different traffic distributions and attack categories.

4.4. 10-Fold Cross-Validation Analysis Under the Controlled Balanced Benchmark

To examine the consistency of the proposed models under the controlled balanced benchmark, 10-FCV is performed on the balanced dataset [24]. The purpose of this analysis is to determine whether the relative performance of the proposed models remains stable across different folds generated from the same balanced experimental distribution. Because the folds contain observations originating from the autoencoder-balanced dataset, these results should not be interpreted as independent evidence of leakage-free real-world generalization. Instead, the analysis measures within-benchmark stability and complements the separately reported split-first, training-only balancing experiment, in which the test observations remain completely untouched by preprocessing and synthetic generation.

Table 11 summarizes the 10-FCV results of the proposed PH-BiLSTM-FCNN, Q-PH-BiLSTM-FCNN, and EAL-PH-BiLSTM-FCNN models across multiple evaluation metrics. In the case of the PH-BiLSTM-FCNN model, the true accuracy of folds is also consistently around 0.845, F1-scores are near 0.844, and the ROC-AUCs are between 0.9641 and 0.9672, which means that classification is quite stable and reliably predictive. The respective values of MCC and CK that are over 0.80 in all folds, also confirm that there is a high level of agreement between the predicted and true labels and further prove the ability to resist the influence of class imbalance.

Table 11. 10-fold cross-validation results across all metrics for proposed models.

Model/ Fold	Acc	Prec	Rec	F1	ROC-AUC	PR-AUC	LogLoss	MCC	CK
PH-BiLSTM-FCNN Model									
Fold 1	0.8426	0.8438	0.8426	0.8419	0.9641	0.8523	0.2514	0.8046	0.8031
Fold 2	0.8463	0.8475	0.8463	0.8454	0.9662	0.8559	0.2468	0.8089	0.8076
Fold 3	0.8441	0.8450	0.8441	0.8430	0.9650	0.8538	0.2497	0.8062	0.8054
Fold 4	0.8478	0.8486	0.8478	0.8469	0.9669	0.8567	0.2459	0.8097	0.8088
Fold 5	0.8439	0.8447	0.8439	0.8428	0.9648	0.8534	0.2503	0.8058	0.8049
Fold 6	0.8455	0.8466	0.8455	0.8446	0.9658	0.8549	0.2476	0.8075	0.8068
Fold 7	0.8469	0.8478	0.8469	0.8459	0.9663	0.8556	0.2462	0.8086	0.8079
Fold 8	0.8434	0.8442	0.8434	0.8423	0.9645	0.8529	0.2508	0.8051	0.8042
Fold 9	0.8481	0.8491	0.8481	0.8472	0.9672	0.8571	0.2453	0.8104	0.8096
Fold 10	0.8447	0.8456	0.8447	0.8438	0.9653	0.8540	0.2491	0.8069	0.8060
Q-PH-BiLSTM-FCNN Model									
Fold 1	0.9138	0.9189	0.9138	0.9126	0.9898	0.9607	0.1504	0.8943	0.8921
Fold 2	0.9162	0.9221	0.9162	0.9151	0.9911	0.9628	0.1476	0.8979	0.8956
Fold 3	0.9149	0.9197	0.9149	0.9138	0.9902	0.9613	0.1493	0.8956	0.8937
Fold 4	0.9176	0.9234	0.9176	0.9166	0.9918	0.9635	0.1462	0.8994	0.8973
Fold 5	0.9151	0.9203	0.9151	0.9140	0.9904	0.9617	0.1490	0.8961	0.8941
Fold 6	0.9183	0.9242	0.9183	0.9174	0.9921	0.9642	0.1458	0.9006	0.8984
Fold 7	0.9144	0.9195	0.9144	0.9133	0.9900	0.9610	0.1498	0.8952	0.8932
Fold 8	0.9169	0.9227	0.9169	0.9158	0.9913	0.9630	0.1471	0.8982	0.8961
Fold 9	0.9178	0.9239	0.9178	0.9168	0.9917	0.9638	0.1464	0.8997	0.8976
Fold 10	0.9156	0.9211	0.9156	0.9145	0.9906	0.9621	0.1487	0.8969	0.8949
EAL-PH-BiLSTM-FCNN Model									
Fold 1	0.8594	0.8821	0.8594	0.8562	0.9756	0.8798	0.2996	0.9098	0.8319
Fold 2	0.8621	0.8859	0.8621	0.8591	0.9772	0.8827	0.2961	0.9136	0.8355
Fold 3	0.8608	0.8836	0.8608	0.8576	0.9763	0.8809	0.2984	0.9112	0.8334
Fold 4	0.8637	0.8872	0.8637	0.8608	0.9779	0.8841	0.2953	0.9151	0.8369
Fold 5	0.8612	0.8841	0.8612	0.8580	0.9766	0.8811	0.2978	0.9123	0.8342
Fold 6	0.8644	0.8883	0.8644	0.8615	0.9783	0.8854	0.2941	0.9164	0.8381
Fold 7	0.8601	0.8830	0.8601	0.8568	0.9760	0.8802	0.2991	0.9104	0.8325
Fold 8	0.8629	0.8864	0.8629	0.8599	0.9771	0.8833	0.2965	0.9142	0.8359
Fold 9	0.8635	0.8871	0.8635	0.8606	0.9776	0.8840	0.2959	0.9150	0.8366
Fold 10	0.8618	0.8848	0.8618	0.8586	0.9767	0.8815	0.2974	0.9127	0.8346

The Q-PH-BiLSTM-FCNN model exhibits the most consistent and superior performance across all folds, with accuracy values tightly clustered between 0.9138 and 0.9183 and F1-scores around 0.915. The ROC-AUC consistently exceeds 0.989, while PR-AUC values remain above 0.960, reflecting exceptional class separability and precision-recall balance. Additionally, the low LL values, approximately 0.146–0.150, indicate highly confident predictions, and the MCC and CK values approaching 0.90 highlight the strong generalization ability achieved through RL-based hyperparameter optimization. In the case of the EAL-PH-BiLSTM-FCNN model, FCV results are stable and reliable with accuracy values of

0.8594–0.8644 and F1-scores of 0.859. The values of ROC–AUC are always above 0.975 and the values of PR–AUC are above 0.879 in all folds, which proves effective discrimination despite less labeled data. The fact that the LL is relatively higher than that of Q-PH-BiLSTM-FCNN shows the uncertainty of active learning in the initial stages, but the value of the MCC and CK above 0.91 and 0.83, respectively, confirms the strength and stability of the EAL strategy. Overall, the 10-FCV results indicate that the relative performance of the proposed models is stable across different partitions of the controlled balanced benchmark. In particular, Q-PH-BiLSTM-FCNN consistently obtains the strongest within-benchmark performance, followed by EAL-PH-BiLSTM-FCNN and PH-BiLSTM-FCNN. However, because these folds originate from the balanced dataset, the observed stability should not be interpreted as proof of leakage-free generalization to independent real-world traffic. Such generalization is examined separately through the split-first sensitivity experiment and external-dataset evaluation.

4.5. Confidence-Interval Analysis Under the Controlled Balanced Benchmark

The 95% Confidence Interval (CI) analysis of the proposed PH-BiLSTM-FCNN, Q-PH-BiLSTM-FCNN, and EAL-PH-BiLSTM-FCNN models based on the key performance metrics is provided in Table 12 and will give a statistical understanding of the stability and reliability of the results obtained. In the case of the PH-BiLSTM-FCNN model, the average value of 0.8453 with a small depth of [0.8440, 0.8467] shows that the model is predictive and consistent when assessed repeatedly. Similar tight intervals are observed for precision, recall, and F1-score, indicating low variance in classification outcomes. The ROC-AUC and PR-AUC confidence ranges also indicate the consistency of discrimination ability, and MCC and CK values with restricted intervals indicate consistent and strong agreement by classes and resistance to class imbalance.

Table 12. Confidence interval (95%) analysis of proposed models across all metrics.

Model	Acc	Prec	Rec	F1	ROC-AUC	PR-AUC	MCC	CK
PH-BiLSTM-FCNN Model								
Mean	0.8453	0.8463	0.8453	0.8444	0.9656	0.8547	0.8074	0.8064
95% CI	[0.8440, 0.8467]	[0.8449, 0.8476]	[0.8440, 0.8467]	[0.8430, 0.8457]	[0.9649, 0.9664]	[0.8535, 0.8558]	[0.8060, 0.8088]	[0.8049, 0.8079]
Q-PH-BiLSTM-FCNN Model								
Mean	0.9161	0.9216	0.9161	0.9150	0.9909	0.9624	0.8974	0.8953
95% CI	[0.9150, 0.9172]	[0.9202, 0.9230]	[0.9150, 0.9172]	[0.9138, 0.9161]	[0.9903, 0.9915]	[0.9615, 0.9633]	[0.8959, 0.8989]	[0.8938, 0.8968]
EAL-PH-BiLSTM-FCNN Model								
Mean	0.8620	0.8853	0.8620	0.8589	0.9769	0.8823	0.9131	0.8350
95% CI	[0.8608, 0.8632]	[0.8838, 0.8867]	[0.8608, 0.8632]	[0.8576, 0.8602]	[0.9763, 0.9775]	[0.8810, 0.8836]	[0.9115, 0.9146]	[0.8335, 0.8364]

Q-PH-BiLSTM-FCNN model is the most statistically stable among all the proposed solutions and the mean accuracy is 0.9161 with a very small confidence intervals of [0.9150, 0.9172]. In line with that, the values of precision, recall, and F1-score are very concentrated, highlighting the consistency achieved with RL-based hyperparameter optimization. The ROC–AUC and PR–AUC confidence interval of [0.9903, 0.9915] and [0.9615, 0.9633], respectively, point to a near-optimal and very reliable class separability. In addition, high predictive agreement is ensured by the narrow MCC and CK intervals, which are focused on 0.8974 and 0.8953. For the EAL-PH-BiLSTM-FCNN model, the mean

accuracy of 0.8620 with a confidence interval of [0.8608, 0.8632] highlights stable generalization despite reduced labeling effort. The intervals of the precision, recall, and F1-score are small enough, which proves the efficiency of EAL in performance consistency. The high ROC-AUC and PR-AUC intervals also verify high discrimination ability despite the fact that its confidence limits are somewhat broader compared to those of Q-PH-BiLSTM-FCNN. Overall, the confidence interval analysis indicates that all proposed models can provide statistically credible results, with Q-PH-BiLSTM-FCNN being the most stable, then EAL-PH-BiLSTM-FCNN and PH-BiLSTM-FCNN.

4.6. Dependence-Corrected Statistical Analysis

The performance differences among the proposed models are statistically examined using the Nadeau–Bengio corrected resampled t -test. Unlike the conventional paired t -test, this method accounts for the dependence introduced by the overlapping training subsets of cross-validation. Since the experiments are performed using one matched 10-FCV run, the correction is applied using a test-to-training-set ratio of 1/9. For each metric, fold-wise paired differences are calculated between Q-PH-BiLSTM-FCNN and PH-BiLSTM-FCNN and between EAL-PH-BiLSTM-FCNN and PH-BiLSTM-FCNN.

The purpose of the present statistical analysis is limited to testing whether the observed performance differences between the proposed models remain consistent across matched folds of the controlled balanced benchmark. Statistical significance cannot correct, remove, or compensate for limitations introduced by the preceding data-generation and balancing protocol. Therefore, the corrected p -values, confidence intervals, corrected t -statistics, and effect sizes reported in this section should not be interpreted as evidence that the balanced folds represent fully independent real-world observations. Instead, these statistics quantify the consistency and magnitude of the relative model differences conditional on the adopted controlled balanced benchmark. The leakage-controlled split-first experiment is reported separately to provide an independent assessment using untouched real test observations.

The corrected test statistic is calculated as

$$t_{\text{corr}} = \frac{\bar{d}}{\sqrt{\left(\frac{1}{kr} + \frac{n_{\text{test}}}{n_{\text{train}}}\right) s_d^2}}, \quad (43)$$

where $\bar{d}$ is the mean paired performance difference, s_d^2 is the sample variance of the fold-wise differences, $k = 10$ is the number of folds, $r = 1$ is the number of cross-validation repetitions, and $n_{\text{test}}/n_{\text{train}} = 1/9$. Mean differences are defined as the candidate-model result minus the PH-BiLSTM-FCNN result. Therefore, positive differences indicate improvement for performance metrics, whereas a negative difference indicates improvement for LL. Holm adjustment is applied across the nine metrics within each model comparison to control the family-wise error rate. Table 13 presents the corrected results.

The corrected results indicate that Q-PH-BiLSTM-FCNN consistently outperforms PH-BiLSTM-FCNN across all evaluated metrics within the controlled balanced benchmark. The mean accuracy improvement is 0.07073, with a corrected 95% confidence interval of [0.06898, 0.07248]. Similar positive differences are observed for precision, recall, F1-score, ROC-AUC, PR-AUC, MCC, and CK. Q-PH-BiLSTM-FCNN also reduces LL by 0.10028, as indicated by the negative mean difference and the confidence interval [−0.10230, −0.09826]. All corresponding Holm-adjusted p -values remain below 0.05, providing dependence-corrected evidence that the relative improvements are consistent across the matched folds. EAL-PH-BiLSTM-FCNN also provides statistically significant improvements over PH-BiLSTM-FCNN in accuracy, precision, recall, F1-score, ROC-AUC, PR-AUC, MCC, and CK

within the controlled balanced benchmark. Its mean accuracy difference is 0.01666, with a corrected 95% confidence interval of [0.01481, 0.01851]. However, EAL-PH-BiLSTM-FCNN increases LL by 0.04871, with a confidence interval of [0.04658, 0.05084], indicating less calibrated or more uncertain probability estimates despite improvements in classification and agreement metrics. This behavior is consistent with the reduced-label learning setting of EAL-PH-BiLSTM-FCNN, in which informative samples are selected iteratively rather than using the complete labeled training set.

Table 13. Nadeau–Bengio dependence-corrected statistical analysis based on the matched 10-fold cross-validation results.

Metric	Mean Difference	Corrected 95% CI	Corrected t	Holm-Adjusted p	Cohen's d_z
Q-PH-BiLSTM-FCNN vs. PH-BiLSTM-FCNN					
Accuracy	0.07073	[0.06898, 0.07248]	91.49	5.65×10^{-14}	42.04
Precision	0.07529	[0.07339, 0.07719]	89.74	5.65×10^{-14}	41.23
Recall	0.07073	[0.06898, 0.07248]	91.49	5.65×10^{-14}	42.04
F1-score	0.07061	[0.06883, 0.07239]	89.51	5.65×10^{-14}	41.13
ROC-AUC	0.02529	[0.02437, 0.02621]	62.28	3.58×10^{-13}	28.62
PR-AUC	0.10775	[0.10632, 0.10918]	170.21	3.82×10^{-16}	78.20
LL	−0.10028	[−0.10230, −0.09826]	−112.20	1.44×10^{-14}	−51.55
MCC	0.09002	[0.08803, 0.09201]	102.48	2.62×10^{-14}	47.09
CK	0.08887	[0.08693, 0.09081]	103.42	2.62×10^{-14}	47.52
EAL-PH-BiLSTM-FCNN vs. PH-BiLSTM-FCNN					
Accuracy	0.01666	[0.01481, 0.01851]	20.35	2.34×10^{-8}	9.35
Precision	0.03896	[0.03693, 0.04099]	43.42	6.37×10^{-11}	19.95
Recall	0.01666	[0.01481, 0.01851]	20.35	2.34×10^{-8}	9.35
F1-score	0.01453	[0.01260, 0.01646]	17.00	3.80×10^{-8}	7.81
ROC-AUC	0.01132	[0.01040, 0.01224]	27.98	1.85×10^{-9}	12.86
PR-AUC	0.02764	[0.02585, 0.02943]	34.94	3.82×10^{-10}	16.05
LL	0.04871	[0.04658, 0.05084]	51.74	1.51×10^{-11}	23.77
MCC	0.10570	[0.10349, 0.10791]	108.18	2.25×10^{-14}	49.70
CK	0.02853	[0.02642, 0.03064]	30.61	1.04×10^{-9}	14.06

The extremely large corrected t -statistics and standardized effect sizes reported for several metrics should be interpreted cautiously. These values are partly caused by the very small fold-to-fold variance of the paired performance differences and therefore should not be interpreted as indicating an equally extreme practical improvement or an unrestricted real-world generalization advantage. For this reason, the revised analysis places greater emphasis on the direction and magnitude of the mean paired differences, the corrected 95% confidence intervals, and the Holm-adjusted p -values, while the standardized effect sizes are retained only as supplementary measures of within-benchmark separation.

Importantly, the statistical analysis presented here is not used to validate or compensate for the balancing-before-splitting protocol. Its purpose is only to examine whether the relative differences among PH-BiLSTM-FCNN, Q-PH-BiLSTM-FCNN, and EAL-PH-BiLSTM-FCNN are reproducible across matched folds of the same controlled balanced benchmark. A separate leakage-controlled sensitivity experiment was therefore conducted using a split-first protocol, in which the original real observations were partitioned before learned preprocessing and autoencoder-based augmentation, and synthetic samples were restricted exclusively to the training partition. The held-out test set in that experiment remained composed of untouched real observations.

4.7. Impact of Alternative Active Learning Techniques on PH-BiLSTM-FCNN Performance

The results reported in Table 14 analyze the impact of applying different active learning strategies to the proposed PH-BiLSTM-FCNN model by comparing Uncertainty-Aware

Learning (UAL) and Monte Carlo (MC) Dropout in terms of predictive performance, reliability, error behavior, and computational efficiency. When UAL is integrated with PH-BiLSTM-FCNN, the model achieves an accuracy of 0.8421, precision of 0.8440, recall of 0.8421, and an F1-score of 0.8404, indicating that uncertainty-driven sample selection enables the model to maintain balanced classification performance even with limited labeled data.

Table 14. Performance comparison of Uncertainty-Aware Learning (UAL) and MC Dropout.

Metric	UAL	MC Dropout
Accuracy	0.8421	0.7514
Precision	0.8440	0.8225
Recall	0.8421	0.7514
F1 Score	0.8404	0.7147
CK	0.8026	0.6893
MCC	0.8044	0.7155
HL	0.1579	0.2486
LL	0.2565	0.4469
ROC-AUC	0.9650	0.9613
PR-AUC	0.8534	0.8437
ET (s)	794.39	3179.25

The respective CK and MCC of 0.8026 and 0.8044 also confirm that there is high concordance between predicted and true labels, which is indicative of high strength against class imbalance. Furthermore, UAL achieves a ROC-AUC value of 0.9650 and PR-AUC of 0.8534, which show a good discriminative performance and good PR trade-off in the presence of skewed intrusion distributions. In terms of errors, the model will have a HL of 0.1579 and LL of 0.2565, which implies rather confident predictions but with moderate uncertainty, and a sensible ET of 794.39 s.

In comparison, the MC Dropout approach shows significantly reduced performance in most of the evaluation metrics. The accuracy drops to 0.7514 with a corresponding F1-score of 0.7147, while precision and recall decrease to 0.8225 and 0.7514, respectively. This is also reflected by the decline in MCC and CK values, which are 0.7155 and 0.6893 respectively, corresponding to decreased classification reliability and reduced inter-class agreement. Even though MC Dropout achieved an ROC-AUC and PR-AUC of 0.9613 and 0.8437 respectively, its larger HL of 0.2486 and significantly higher LL of 0.4469 show that its predictions are more uncertain and the misclassification rates are higher. In addition, the ET grows exponentially to 3179.25 s, showing the high computational cost of using several stochastic forward passes in the process of inference.

When compared with the EAL-PH-BiLSTM-FCNN results reported in Table 4, both UAL and MC Dropout-based variants exhibit inferior performance across nearly all metrics. The EAL-PH-BiLSTM-FCNN achieves higher accuracy of 0.8616, F1-score of 0.8584, MCC of 0.9120, and lower LL of 0.2977, indicating more reliable and confident predictions. The performance difference can be explained by the EAL mechanism used in EAL-PH-BiLSTM-FCNN that directly quantifies predictive uncertainty with the complete class probability distribution and biases samples around decision boundaries. Conversely, UAL generally uses more basic measures of uncertainty that might not effectively represent multi-class ambiguity, whereas MC Dropout estimates Bayesian uncertainty by stochastic sampling, introducing noise and uncertainty estimation instability in high-dimensional

tabular data. As a result, the limitations result in suboptimal sample selection, slower convergence and lower generalization performance relative to the EAL strategy.

4.8. Label-Efficient Active Learning Analysis

To provide a robust assessment of the proposed EAL strategy, the comparison with random sampling is repeated across five independent random seeds, {42, 52, 62, 72, 82}. Each seed generated a different initial stratified labeled subset, while EAL and random sampling are initialized from exactly the same labeled and unlabeled pools within the corresponding run. Both acquisition strategies are evaluated under identical model architecture, optimization settings, training epochs, and labeling budgets of 10%, 20%, 30%, 40%, 50%, and 60%. This paired experimental design reduces the possibility that differences between the acquisition strategies arise solely from a favorable initial labeled subset.

Performance at the intermediate labeling budgets is measured exclusively on a fixed validation partition. The held-out test partition is not accessed during uncertainty estimation, sample acquisition, learning-curve construction, or labeling-budget analysis. For each strategy and labeling budget, the reported results are averaged over the five independent runs. Table 15 reports the mean and standard deviation, whereas Figure 9 additionally visualizes the corresponding 95% confidence intervals. The 100% labeled validation reference shown in the figures represents the fully supervised model trained using all available labeled training observations and is included only as a reference level for interpreting label efficiency.

Table 15 demonstrates that the relative performance of the two acquisition strategies varies across labeling budgets. Both strategies have identical mean performance at the initial 10% budget because, within each seed, they begin from exactly the same labeled subset. EAL subsequently achieves higher mean accuracy at the 20%, 30%, 40%, and 50% budgets, with corresponding improvements in F1-score and MCC. For example, at the 30% budget, EAL obtains mean accuracy, F1-score, and MCC values of 0.8881, 0.8803, and 0.8688, respectively, compared with 0.8509, 0.8326, and 0.8312 for random sampling. At 50%, the corresponding entropy-based values are 0.8667, 0.8657, and 0.8349, whereas random sampling achieves 0.8007, 0.7911, and 0.7594.

Table 15. Multi-seed validation performance of entropy-based and random sampling across different labeling budgets. Values are reported as mean $\pm$ standard deviation over five independent runs.

Metric	10%	20%	30%	40%	50%	60%
Entropy Sampling						
Accuracy	0.8340 $\pm$ 0.0340	0.8227 $\pm$ 0.0748	0.8881 $\pm$ 0.0466	0.8540 $\pm$ 0.0320	0.8667 $\pm$ 0.0611	0.8500 $\pm$ 0.0649
F1-Score	0.8073 $\pm$ 0.0504	0.8017 $\pm$ 0.0881	0.8803 $\pm$ 0.0577	0.8449 $\pm$ 0.0413	0.8657 $\pm$ 0.0619	0.8348 $\pm$ 0.0789
MCC	0.8139 $\pm$ 0.0314	0.7953 $\pm$ 0.0851	0.8688 $\pm$ 0.0504	0.8282 $\pm$ 0.0352	0.8349 $\pm$ 0.0757	0.8261 $\pm$ 0.0731
ROC-AUC	0.9693 $\pm$ 0.0178	0.9752 $\pm$ 0.0107	0.9808 $\pm$ 0.0150	0.9811 $\pm$ 0.0169	0.9813 $\pm$ 0.0116	0.9772 $\pm$ 0.0148
PR-AUC	0.8914 $\pm$ 0.0557	0.9024 $\pm$ 0.0387	0.9208 $\pm$ 0.0557	0.9212 $\pm$ 0.0613	0.9187 $\pm$ 0.0480	0.9041 $\pm$ 0.0574
Random Sampling						
Accuracy	0.8340 $\pm$ 0.0340	0.7919 $\pm$ 0.0229	0.8509 $\pm$ 0.0425	0.8389 $\pm$ 0.0863	0.8007 $\pm$ 0.0713	0.8821 $\pm$ 0.0638
F1-Score	0.8073 $\pm$ 0.0504	0.7815 $\pm$ 0.0314	0.8326 $\pm$ 0.0531	0.8272 $\pm$ 0.0952	0.7911 $\pm$ 0.0814	0.8783 $\pm$ 0.0668
MCC	0.8139 $\pm$ 0.0314	0.7492 $\pm$ 0.0254	0.8312 $\pm$ 0.0456	0.8089 $\pm$ 0.1022	0.7594 $\pm$ 0.0832	0.8571 $\pm$ 0.0773
ROC-AUC	0.9693 $\pm$ 0.0178	0.9698 $\pm$ 0.0060	0.9809 $\pm$ 0.0089	0.9831 $\pm$ 0.0094	0.9754 $\pm$ 0.0100	0.9852 $\pm$ 0.0071
PR-AUC	0.8914 $\pm$ 0.0557	0.8804 $\pm$ 0.0270	0.9185 $\pm$ 0.0330	0.9350 $\pm$ 0.0319	0.8967 $\pm$ 0.0410	0.9359 $\pm$ 0.0297

However, the results do not indicate uniform dominance of EAL acquisition. At the 60% validation budget, random sampling produces higher mean accuracy, F1-score, and MCC values of 0.8821, 0.8783, and 0.8571, respectively, compared with 0.8500, 0.8348,

and 0.8261 for EAL. A similar budget-dependent pattern is observed for the ranking metrics. EAL produces higher mean PR-AUC at the 20%, 30%, and 50% budgets, whereas random sampling is higher at 40% and 60%. These results therefore support an interpretation based on the complete learning trajectory rather than on a single favorable operating point.

Figure 9 further illustrates the variability associated with the initial labeled subset and acquisition strategy. Figure 9a shows that EAL generally maintains a higher mean accuracy through the intermediate 20–50% budget range, whereas random sampling improves substantially at the 60% budget. A similar pattern is visible for F1-score and MCC in Figure 9b,c. Importantly, the confidence bands overlap at several labeling budgets, demonstrating that the magnitude and even direction of the difference between acquisition strategies can depend on the initial labeled set. Consequently, individual budget-level differences should not be interpreted as evidence of universal superiority. The PR-AUC curves in Figure 9d provide an additional class-sensitive view of the acquisition behavior. Both methods maintain relatively high PR-AUC values, but their learning curves cross at different budgets. This finding is particularly relevant for intrusion detection because PR-AUC is sensitive to the quality of positive-class ranking and is therefore informative when class frequencies are unequal. Taken together, the four learning curves demonstrate that EAL acquisition is competitive across the evaluated budgets, while its benefit is dependent on both the labeling budget and initialization.

To avoid conclusions based on isolated labeling budgets, the two acquisition strategies are additionally compared using the normalized Area Under the Learning Curve (AULC). The AULC is independently calculated for every random seed across the complete 10–60% labeling trajectory. The resulting paired AULC values are then compared using a two-sided Wilcoxon signed-rank test. This analysis directly evaluates the overall learning behavior rather than selecting the best-performing budget from either strategy.

As reported in Table 16, EAL achieves higher mean AULC values for all reported classification metrics. The largest descriptive improvements are observed for CK, MCC, F1-score, and accuracy. Specifically, EAL increases the mean AULC from 0.8281 to 0.8547 for accuracy, from 0.8150 to 0.8427 for F1-score, and from 0.7968 to 0.8295 for MCC. These results indicate that EAL acquisition provides a higher average classification trajectory across the complete set of evaluated labeling budgets rather than merely at one selected operating point. Nevertheless, the paired Wilcoxon analysis does not establish statistical superiority at the conventional $\alpha = 0.05$ significance level. Accuracy, recall, F1-score, CK, and MCC each yield $p = 0.0625$, while the differences for ROC-AUC and PR-AUC are smaller, with $p = 0.8125$ and $p = 0.6250$, respectively. Therefore, the analysis does not support a claim that EAL is consistently or statistically superior to random sampling. Instead, the results indicate a descriptive overall advantage for EAL across several metrics, accompanied by non-negligible variability across initial labeled subsets. This interpretation directly accounts for the complete learning curves and avoids drawing conclusions from isolated labeling-budget results.

After completing the learning-curve analysis, the final labeling budget is fixed at 60% before the held-out test partition is accessed. Each finalized model is then retrained using the corresponding enriched labeled training set and evaluated on the previously untouched test partition. The resulting multi-seed test performance is summarized in Table 17.

At the fixed 60% labeling budget, EAL obtains a higher mean held-out test accuracy of 0.8961 compared with 0.8160 for random sampling, while the corresponding F1-scores are 0.8951 and 0.7960. EAL also achieves a higher mean MCC of 0.8720 and lower LL of 0.2174, compared with 0.7867 and 0.3021 for random sampling. Similarly, the mean ROC-AUC and PR-AUC values increase from 0.9725 and 0.8956 under random sampling to 0.9836 and 0.9304 under EAL acquisition. These results provide evidence that the finalized

entropy-acquired models can achieve strong held-out performance at the predetermined labeling budget; however, the comparatively large standard deviations for several EAL metrics also confirm that performance remains sensitive to the initial labeled subset.

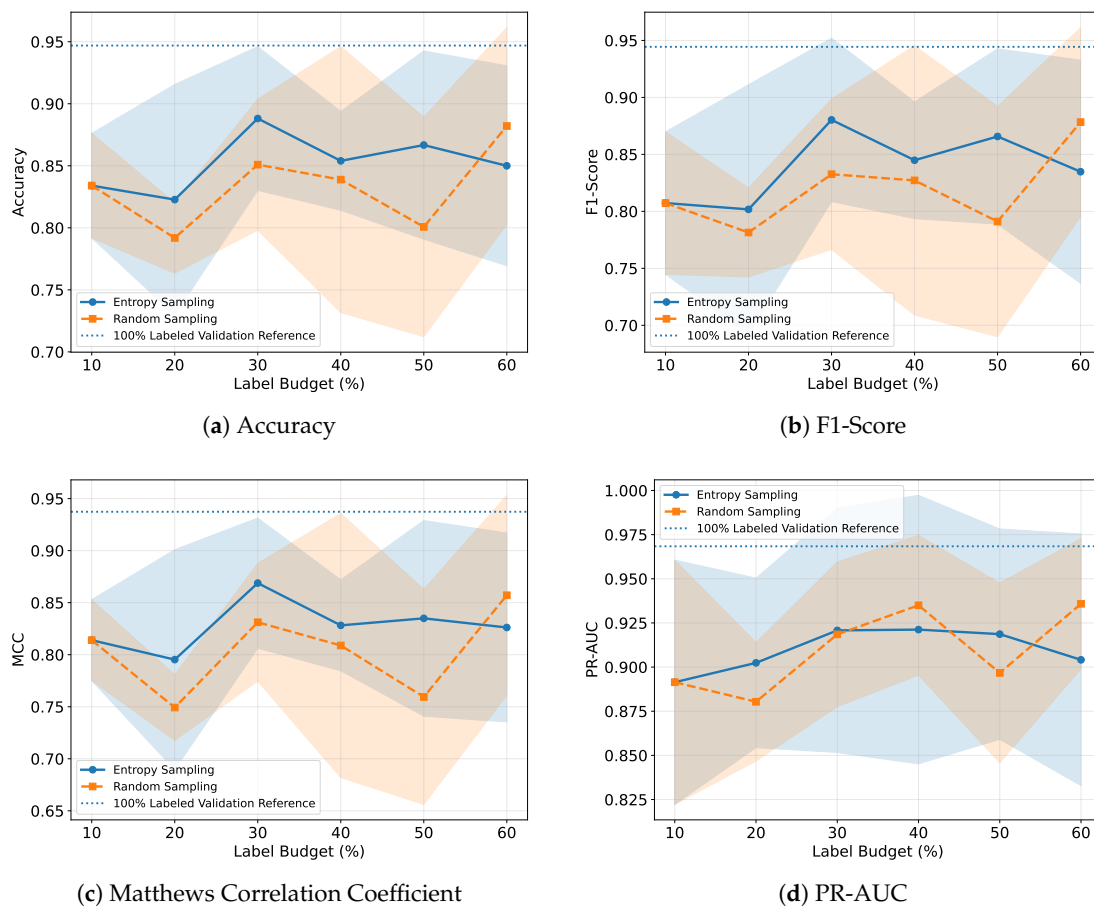


Figure 9. Multi-seed learning-curve comparison of entropy-based and random sampling under different labeling budgets. The curves represent mean performance across five independent initial labeled sets, the shaded regions denote the corresponding 95% confidence intervals, and the horizontal dotted reference represents validation performance obtained using the fully labeled training set.

Table 16. Complete learning-curve comparison of entropy-based and random sampling using normalized AULC and paired Wilcoxon analysis.

Metric	Entropy AULC	Random AULC	Mean Difference [95% CI]	<i>p</i> -Value
Accuracy	0.8547	0.8281	0.0266 [0.0039, 0.0494]	0.0625
Precision	0.8780	0.8578	0.0202 [−0.0009, 0.0412]	0.1250
Recall	0.8547	0.8281	0.0266 [0.0039, 0.0494]	0.0625
F1-Score	0.8427	0.8150	0.0277 [0.0073, 0.0481]	0.0625
CK	0.8184	0.7851	0.0333 [0.0049, 0.0617]	0.0625
MCC	0.8295	0.7968	0.0326 [0.0023, 0.0629]	0.0625
ROC-AUC	0.9783	0.9773	0.0010 [−0.0044, 0.0065]	0.8125
PR-AUC	0.9122	0.9088	0.0033 [−0.0150, 0.0216]	0.6250

Overall, the multi-seed analysis provides a more conservative assessment than the original single-run experiment. EAL acquisition demonstrates higher average complete-curve performance for several classification metrics and stronger mean final test performance at the fixed 60% labeling budget, but neither acquisition strategy dominates every intermediate labeling budget. Moreover, the paired complete-curve statistical comparisons do not reach the $p < 0.05$ significance threshold. Accordingly, the findings characterize EAL as a

competitive and potentially label-efficient acquisition strategy rather than claiming that it is universally preferable to random sampling.

Table 17. Final held-out test performance at the fixed 60% labeling budget. Values are reported as mean $\pm$ standard deviation over five independent runs.

Metric	Entropy Sampling	Random Sampling
Accuracy	0.8961 $\pm$ 0.0752	0.8160 $\pm$ 0.0293
Precision	0.9013 $\pm$ 0.0755	0.8540 $\pm$ 0.0431
Recall	0.8961 $\pm$ 0.0752	0.8160 $\pm$ 0.0293
F1-Score	0.8951 $\pm$ 0.0761	0.7960 $\pm$ 0.0358
MCC	0.8720 $\pm$ 0.0936	0.7867 $\pm$ 0.0391
LL	0.2174 $\pm$ 0.1006	0.3021 $\pm$ 0.0638
ROC-AUC	0.9836 $\pm$ 0.0155	0.9725 $\pm$ 0.0104
PR-AUC	0.9304 $\pm$ 0.0628	0.8956 $\pm$ 0.0448

4.9. Performance Analysis of PH-BiLSTM-FCNN with Different Reinforcement Learning and Optimization Techniques

To ensure a fair comparison, Q-learning, random search, and Bayesian optimization are evaluated under an identical optimization budget. Each method is allowed the same number of 10 hyperparameter-evaluation trials, uses the same PH-BiLSTM-FCNN architecture, training and validation partitions, candidate hyperparameter ranges, number of training epochs per trial, and evaluation metric for model selection. The test set is not used during hyperparameter optimization and is accessed only once to evaluate the final configuration selected by each optimization strategy. Table 18 presents a comparative evaluation of the proposed PH-BiLSTM-FCNN model when optimized using two alternative RL strategies, namely PPO and DDPG. When PPO is used to optimize the hyperparameters, the PH-BiLSTM-FCNN+PPO set-up has an accuracy of 0.9018, precision of 0.9073, recall of 0.9018 and an F1-score of 0.8994, which is significantly higher than the non-optimized PH-BiLSTM-FCNN baseline. Good class separability and a good PR balance are also demonstrated by the discriminative capability of the model with a high ROC-AUC of 0.9872 and a PR-AUC of 0.9485. When it comes to the error behavior, the model is characterized by a decreased LL of 0.1724 and a HL of 0.0982, which is indicative of confident predictions and fewer misclassifications. The related MCC and CK values of 0.8786 and 0.8759 indicate strong classification congruency and reliability, at the expense of a rather high ET of 1826.43 s, as PPO is on-policy and the policy is updated on a regular basis.

Conversely, the PH-BiLSTM-FCNN+DDPG model exhibits relatively poorer results on most measures. Accuracy is reduced to 0.8896 and the F1-score is 0.8861, whereas the precision and recall have been reduced to 0.8942 and 0.8896, respectively. This degradation is also indicated by lower ROC-AUC and PR-AUC values of 0.9826 and 0.9368, which means that the discriminative capacity of the material is not as good as PPO. Measures of error also indicate such a decrease, with an increased LL of 0.1969 and HL of 0.1104, indicating more uncertainty in predictions and misclassification. The MCC and CK of 0.8614 and 0.8587 prove the low robustness and agreement among classes. Also, the ET grows to 2049.88 s, demonstrating the computational cost of continuous action-space exploration and unstable policy convergence of DDPG in discrete hyperparameter optimization problems.

When compared with the Q-PH-BiLSTM-FCNN results reported in Table 4, both PPO and DDPG-based optimization strategies exhibit inferior performance across all evaluation metrics. The Q-PH-BiLSTM-FCNN achieves a higher accuracy of 0.9154, F1-score of 0.9144, ROC-AUC of 0.9905, PR-AUC of 0.9619, and significantly lower LL of 0.1489, alongside superior MCC and CK values of 0.8962 and 0.8942, respectively. The performance difference can be explained by the fact that Q-learning is inherently suited to discrete and low-

dimensional action spaces, including the choice of learning rate and large-scale batch sizes. Conversely, PPO and DDPG are specifically tailored to continuous or high-dimensional control problems and add extra policy approximation noise and variance to discrete hyperparameter tuning. Such a misfit results in an inefficient exploration–exploitation ratio, slower convergence, and lower stability during training. Therefore, both PPO and DDPG improve the efficacy of the PH-BiLSTM-FCNN over the unoptimized base. Q-learning is the most successful RL approach to hyperparameter optimization in the proposed Q-PH-BiLSTM-FCNN model.

Table 18. Performance comparison of PPO, DDPG, random-search, and Bayesian-optimized PH-BiLSTM-FCNN models on the final test set.

Metric	PPO	DDPG	RS	BO
Accuracy	0.9018	0.8896	0.8017	0.8553
Precision	0.9073	0.8942	0.8920	0.8888
Recall	0.9018	0.8896	0.8017	0.8553
F1-score	0.8994	0.8861	0.7507	0.8422
ROC-AUC	0.9872	0.9826	0.9538	0.9706
PR-AUC	0.9485	0.9368	0.8073	0.8809
LL	0.1724	0.1969	0.3164	0.2707
MCC	0.8786	0.8614	0.7871	0.8338
CK	0.8759	0.8587	0.7522	0.8191
HL	0.0982	0.1104	0.1983	0.1447

The Bayesian-optimized PH-BiLSTM-FCNN demonstrates stronger overall performance than the random search. It improves accuracy from 0.8017 to 0.8553, weighted recall from 0.8017 to 0.8553, and weighted F1-score from 0.7507 to 0.8422. It also achieves higher ROC-AUC and PR-AUC values of 0.9706 and 0.8809, respectively, compared with 0.9538 and 0.8073 for random search. Moreover, the increases in CK from 0.7522 to 0.8191 and MCC from 0.7871 to 0.8338 indicate better agreement and classification reliability, as listed in Table 18. The Bayesian approach also reduces HL from 0.1983 to 0.1447 and LL from 0.3164 to 0.2707, indicating fewer classification errors and more confident predictions. However, random search obtains a slightly higher weighted precision of 0.8920 compared with 0.8888. Random search evaluates hyperparameter configurations without using information from previous trials, which may result in inefficient exploration and suboptimal parameter selection. In contrast, Bayesian optimization provides better overall results by directing the search toward promising configurations, but its effectiveness depends on the predefined search space and optimization budget and may introduce additional computational overhead.

4.10. Performance Analysis Using Different Data Balancing Techniques

Table 19 summarizes the performance of the proposed PH-BiLSTM-FCNN, Q-PH-BiLSTM-FCNN, and EAL-PH-BiLSTM-FCNN models when Proximity-Weighted Random Affine Shadow Sampling (ProWRAS) and CTGAN are employed for data balancing. Under ProWRAS, the PH-BiLSTM-FCNN model attains an accuracy of 0.8124 with precision, recall, and F1-score values of 0.8187, 0.8124, and 0.8091, respectively. The corresponding ROC-AUC and PR-AUC values of 0.9486 and 0.8214 indicate moderate class separability, while the MCC and CK values of 0.7716 and 0.7684 reflect reduced classification reliability. The Q-PH-BiLSTM-FCNN model shows comparatively stronger performance under ProWRAS, achieving an accuracy of 0.8841 and an F1-score of 0.8815, along with improved ROC-AUC and PR-AUC values of 0.9732 and 0.9127; however, increased LL of 0.1928 and HL of 0.1159 suggest higher prediction uncertainty. For EAL-PH-BiLSTM-FCNN, the accuracy decreases to 0.8369 with an F1-score of 0.8327, and the elevated LL of 0.3384

indicates that uncertainty-based sample selection becomes less effective when trained on affine-transformed synthetic samples.

When CTGAN is applied, a further decline in performance is observed across all proposed models. The PH-BiLSTM-FCNN model records an accuracy of 0.7983 and an F1-score of 0.7937, accompanied by ROC-AUC and PR-AUC values of 0.9412 and 0.8046, and higher log loss of 0.3496, reflecting unstable predictions. Similarly, Q-PH-BiLSTM-FCNN achieves an accuracy of 0.8692 with an F1-score of 0.8654, while MCC and CK decrease to 0.8426 and 0.8391, indicating weakened robustness against class imbalance. The EAL-PH-BiLSTM-FCNN model follows the same trend, with accuracy and F1-score values of 0.8215 and 0.8169 and increased log loss of 0.3659, highlighting the negative impact of GAN-induced distributional noise on entropy-based learning.

Table 19. Performance comparison of proposed models using different data balancing techniques.

Metric	PH-BiLSTM-FCNN	Q-PH-BiLSTM-FCNN	EAL-PH-BiLSTM-FCNN
ProWRAS			
Accuracy	0.8124	0.8841	0.8369
Precision	0.8187	0.8896	0.8423
Recall	0.8124	0.8841	0.8369
F1-Score	0.8091	0.8815	0.8327
ROC-AUC	0.9486	0.9732	0.9624
PR-AUC	0.8214	0.9127	0.8541
LL	0.3129	0.1928	0.3384
MCC	0.7716	0.8613	0.8128
CK	0.7684	0.8589	0.8071
HL	0.1876	0.1159	0.1631
ET (s)	684.52	1294.88	298.41
CTGAN			
Accuracy	0.7983	0.8692	0.8215
Precision	0.8049	0.8741	0.8282
Recall	0.7983	0.8692	0.8215
F1-Score	0.7937	0.8654	0.8169
ROC-AUC	0.9412	0.9678	0.9561
PR-AUC	0.8046	0.9013	0.8427
LL	0.3496	0.2174	0.3659
MCC	0.7542	0.8426	0.7935
CK	0.7489	0.8391	0.7872
HL	0.2017	0.1308	0.1785
ET (s)	702.83	1348.92	321.76

Overall, although both ProWRAS and CTGAN partially alleviate class imbalance, their inability to preserve complex nonlinear feature dependencies leads to degraded performance across accuracy, F1-score, ROC-AUC, MCC, and error-based metrics. In contrast, the autoencoder-based balancing strategy demonstrates superior effectiveness by generating class-consistent synthetic samples, thereby maintaining stronger generalization and more stable learning behavior for the proposed models.

4.11. Component-Wise Ablation Analysis

The component-wise ablation study in Table 20 is conducted using the same primary experimental pipeline adopted for the principal comparative results of this study, namely preprocessing, autoencoder-based class balancing, and subsequent stratified data splitting. Each reduced variant is therefore evaluated under exactly the same data-processing and evaluation conditions as the complete PH-BiLSTM-FCNN model. The purpose of this analysis is to examine the relative contribution of individual architectural and data-balancing components within the primary experimental setting. A separate leakage-controlled sensitivity evaluation is reported independently, in which real observations are partitioned before learned preprocessing and autoencoder-based augmentation and synthetic samples are restricted exclusively to the training partition. As reported in that experiment, the leakage-controlled setting produces saturated or near-perfect performance across the evaluated models, thereby providing limited discriminatory power for identifying the relative contribution of individual architectural components. Consequently, the present ablation results should be interpreted only within the primary evaluation pipeline and not as evidence of leakage-free or cross-environment generalization.

Table 20. Component-wise ablation analysis of the proposed PH-BiLSTM-FCNN framework.

Metric	Ablation Variants				
	Without BiLSTM Encoder	Without Decoder LSTM	Without Single-Position Weighting	Without Learned Fusion	Without Balancing
Accuracy	0.7734	0.9999	0.9999	0.9999	0.3392
Precision	0.6769	0.9999	0.9999	0.9999	0.1845
Recall	0.7744	0.9999	0.9999	0.9999	0.3396
F1-score	0.7073	0.9999	0.9999	0.9999	0.2173
ROC-AUC	0.9700	1.0000	1.0000	1.0000	0.5845
PR-AUC (Macro)	0.8694	1.0000	1.0000	1.0000	0.3599
CK	0.7169	0.9999	0.9998	0.9998	0.1744
MCC	0.7567	0.9999	0.9998	0.9998	0.2716
HL	0.2266	0.0001	0.0001	0.0001	0.6608
LL	0.3442	0.0006	0.0028	0.0028	5.1780

Table 20 demonstrates that removing different components affects both the learning capability and reliability of the proposed PH-BiLSTM-FCNN framework. At the same time, the unusually high scores obtained by several reduced variants require cautious interpretation. In particular, saturated performance after removing specific architectural components may indicate that highly discriminative patterns remain available to the classifier under the primary evaluation setting. Such behavior may arise from strong benchmark separability, correlations among observations, synthetic class-specific patterns, or shortcut features. Therefore, a high score for an ablated variant is not interpreted as evidence that the removed component is unnecessary.

When the single-step BiLSTM encoder is removed, the accuracy decreases to 0.7734, while the precision, recall, and F1-score decline to 0.6769, 0.7744, and 0.7073, respectively. The MCC and CK also decrease to 0.7567 and 0.7169, whereas the HL increases to 0.2266. The single-step BiLSTM encoder constitutes the recurrent nonlinear representation pathway of PH-BiLSTM-FCNN and transforms the complete tabular observation into a learned latent representation before subsequent recurrent refinement. Its removal therefore eliminates this complementary representation pathway and forces the classifier to rely more strongly on the FCNN branch. Although the ROC-AUC remains high at 0.9700, the lower precision and F1-score indicate that the reduced architecture provides less reliable final class discrimination than the complete representation pathway under the primary evaluation setting.

Removing the decoder LSTM produces an apparently saturated accuracy and F1-score of 0.9999, together with ROC-AUC and PR-AUC values of 1.0000. This result should not be interpreted as evidence that the decoder LSTM is intrinsically unnecessary. In the complete recurrent pathway, the decoder LSTM further transforms the output produced by the single-step BiLSTM encoder before the recurrent-branch representation is combined with the FCNN pathway. Removing this stage simplifies the recurrent branch and may allow the classifier to exploit highly separable patterns available elsewhere in the representation. The extremely low LL of 0.0006 and HL of 0.0001 indicate highly confident predictions. Rather than establishing an architectural advantage for the reduced model, this saturated behavior suggests that the underlying benchmark may be sufficiently separable under the primary experimental pipeline for a simpler representation to achieve almost perfect discrimination.

A similar behavior is observed when the single-position normalized weighting operation is removed. In the implemented recurrent pathway, the complete feature vector is represented as one sequence position, so normalization is performed over a sequence length of one. Consequently, the normalized coefficient is mathematically equal to one and should not be interpreted as conventional temporal attention, or feature-level attention. Removing this operation therefore produces only a limited structural change to the effective recurrent representation. The resulting accuracy and F1-score of 0.9999 and ROC-AUC and PR-AUC values of 1.0000 are consistent with the limited functional role of this single-position weighting operation and with the strong separability observed under the primary evaluation setting. These results are not used to claim an attention-related advantage or disadvantage.

Removing the learned fusion mechanism also produces saturated results, including an accuracy and F1-score of 0.9999 and a LL of 0.0028. In the complete PH-BiLSTM-FCNN architecture, the learned fusion layer integrates the 32-dimensional FCNN representation with the class-probability representation produced by the recurrent branch. Removing this layer eliminates the trainable transformation that normally combines information from the two parallel pathways. However, the near-perfect performance of the reduced variant indicates that, under the primary experimental setting, sufficiently discriminative information remains available even without the complete learned fusion operation. Accordingly, the result is interpreted as evidence of limited discriminatory difficulty in this particular evaluation setting rather than as proof that learned fusion is unnecessary.

The similar saturated behavior observed after removing the decoder LSTM, single-position weighting, and learned fusion mechanism further supports the need for cautious interpretation of the ablation results. If several structurally different reduced architectures achieve nearly identical near-perfect performance, the observed scores cannot by themselves establish strong component-level necessity. They instead suggest that classification within the primary pipeline may be dominated by highly discriminative benchmark characteristics or synthetic patterns. The most substantial degradation occurs when autoencoder-based balancing is removed. Accuracy decreases to 0.3392, precision to 0.1845, recall to 0.3396, and F1-score to 0.2173. In addition, ROC-AUC and PR-AUC fall to 0.5845 and 0.3599, while HL and LL increase to 0.6608 and 5.1780, respectively. These results indicate that severe class imbalance substantially affects minority-class learning under the primary experimental pipeline and that the balancing mechanism has a strong influence on the resulting predictive performance.

Nevertheless, the magnitude of this improvement must also be interpreted cautiously. Because the primary pipeline performs autoencoder-based augmentation before the final stratified split, synthetic class-specific structure may contribute to the separability observed by the classifier. For this reason, the ablation result is not presented as independent evidence that synthetic balancing guarantees superior generalization. Instead, it quantifies the

dependence of the proposed model on the balancing stage within the original experimental pipeline. The separately reported leakage-controlled sensitivity evaluation is used to examine behavior when augmentation is restricted exclusively to the training partition.

4.12. Comparison with Strong Tabular Baselines

To provide a more comprehensive assessment of the proposed framework against strong tabular-learning approaches, additional experiments are conducted using eXtreme Gradient Boosting (XGBoost), Categorical Boosting (CatBoost), Light Gradient Boosting Machine (LightGBM), RF, and a validation-tuned Multi-Layer Perceptron (MLP). These models are evaluated using the same data partitions, preprocessing conditions, computing environment, and evaluation metrics employed in the corresponding comparative experiments. The results are reported in Table 21. XGBoost, CatBoost, LightGBM, and RF achieve saturated predictive performance, with accuracy, precision, recall, F1-score, CK, MCC, ROC-AUC, and PR-AUC values equal to 1.0000. Their HL is 0.0000, while their LL values are also approximately zero. The tuned MLP similarly achieves strong predictive performance, obtaining an accuracy of 0.9630, precision of 0.9687, recall of 0.9630, F1-score of 0.9626, CK of 0.9537, MCC of 0.9553, and HL of 0.0370. Although its ROC-AUC and PR-AUC reach 1.0000, its LL of 0.2042 indicates comparatively less confident probability estimates than the tree-based ensemble models.

Table 21. Predictive and computational performance of strong tabular baselines.

Metric	XGBoost	CatBoost	LightGBM	RF	Tuned MLP
Accuracy	1.0000	1.0000	1.0000	1.0000	0.9630
Precision	1.0000	1.0000	1.0000	1.0000	0.9687
Recall	1.0000	1.0000	1.0000	1.0000	0.9630
F1-score	1.0000	1.0000	1.0000	1.0000	0.9626
CK	1.0000	1.0000	1.0000	1.0000	0.9537
MCC	1.0000	1.0000	1.0000	1.0000	0.9553
HL	0.0000	0.0000	0.0000	0.0000	0.0370
LL	0.0000	0.0001	0.0000	0.0000	0.2042
ROC-AUC	1.0000	1.0000	1.0000	1.0000	1.0000
PR-AUC	1.0000	1.0000	1.0000	1.0000	1.0000
ET (s)	208.0896	262.5471	142.9236	288.7082	1693.6105

The exceptionally high performance of the tree-based classifiers is reported transparently and demonstrates that the considered benchmark contains highly discriminative tabular patterns that can be effectively exploited by ensemble decision-tree methods. Therefore, the proposed PH-BiLSTM-FCNN, Q-PH-BiLSTM-FCNN, and EAL-PH-BiLSTM-FCNN models are not claimed to outperform XGBoost, CatBoost, LightGBM, or RF in standalone predictive accuracy under this particular evaluation setting. Instead, the simultaneous near-perfect performance of several structurally different classifiers suggests that the benchmark is highly separable under the adopted evaluation protocol. Consequently, differences in predictive accuracy alone may provide limited discriminatory evidence regarding the superiority of one model architecture over another. This observation is also consistent with the leakage-controlled sensitivity analysis, where multiple models exhibit saturated or near-saturated predictive performance.

The saturated results should nevertheless be interpreted cautiously when considering operational generalization. High performance on a controlled IIoT benchmark does not automatically establish equivalent behavior on previously unseen devices, independent traffic-capture sessions, different temporal periods, changing network configurations, novel attack implementations, or external operational environments. Tree-based ensemble models are particularly effective at identifying highly discriminative thresholds and interactions within structured datasets, which is advantageous for tabular classification

but may also enable them to exploit benchmark-specific correlations when such patterns are present. Accordingly, the present results demonstrate strong in-dataset predictive separability rather than definitive evidence of cross-device, cross-session, temporal, or cross-network generalization. More restrictive device-aware, session-aware, chronological, group-disjoint, and spatiotemporal evaluations are therefore identified as important directions for further investigation.

The computational results additionally reveal important differences among the compared tabular approaches. LightGBM achieves the lowest ET of 142.9236 s, followed by XGBoost 3.4.1 at 208.0896 s, CatBoost 1.2.10 at 262.5471 s, and RF at 288.7082 s. The tuned MLP requires substantially greater computational effort, with an overall tuning and training time of 1693.6105 s because several candidate neural configurations are evaluated using the validation partition before the final model is fixed. During inference, CatBoost provides the lowest ET of 0.2151 s, followed by RF at 2.5848 s and XGBoost at 2.9883 s. The tuned MLP requires 7.7358 s, whereas LightGBM records an inference time of 15.4667 s in the present implementation. These results demonstrate that the most efficient model during training is not necessarily the most efficient during inference and that predictive performance, training cost, and deployment-time computational requirements should be considered jointly when selecting an intrusion-detection model.

The motivation of the proposed framework therefore extends beyond maximizing classification accuracy on a highly separable benchmark. PH-BiLSTM-FCNN investigates the complementary integration of FCNN-based nonlinear feature abstraction and a single-step BiLSTM-based nonlinear representation pathway within a unified hybrid architecture. Q-PH-BiLSTM-FCNN further introduces state-dependent sequential hyperparameter optimization, in which learning-rate and batch-size decisions are guided by validation performance, performance variation, predictive uncertainty, and optimization progress. EAL-PH-BiLSTM-FCNN addresses the additional objective of reducing annotation requirements through uncertainty-guided active learning. Furthermore, the blockchain-assisted component provides an auditable record of optimization and evaluation activities within the proposed framework. These mechanisms address representation learning, sequential optimization, label efficiency, and auditability objectives that are not inherently provided by the conventional tabular classifiers in their standard configurations.

Accordingly, the strong tabular baselines are included as competitive predictive references rather than being used to establish that the proposed architecture achieves the highest classification accuracy in every experimental condition. Their superior benchmark-level predictive scores are acknowledged explicitly. At the same time, the proposed framework is evaluated according to a broader set of objectives involving hybrid representation learning, adaptive hyperparameter optimization, annotation efficiency, computational behavior, and auditable model operation. The inclusion of XGBoost, CatBoost, LightGBM, RF, and tuned MLP therefore provides a more complete and transparent comparative evaluation while also highlighting the need to interpret saturated benchmark scores together with generalization, computational cost, and deployment-oriented considerations.

4.13. Leakage-Controlled Split-First Sensitivity Analysis

To directly address the possibility that preprocessing and synthetic data generation before the final train-test split may influence the reported performance, an additional leakage-controlled sensitivity experiment is conducted. In this experiment, the original real observations are first divided into training and test partitions using an 80:20 stratified split. All learned preprocessing operations and class-specific autoencoder-based augmentation are subsequently restricted exclusively to the training partition. The held-out test partition is not used during normalization, autoencoder training, synthetic sample generation, model

training, hyperparameter optimization, or active sample selection. Consequently, the final test set contains only original real observations that remain completely untouched during model development.

This protocol differs from the primary Balancing → Splitting → Training benchmark, in which autoencoder-based balancing is performed before the final stratified partition. The purpose of the present sensitivity experiment is therefore specifically to examine model behavior under a stricter data-independence condition. However, because the original WUSTL-IIoT-2021 subset is extremely imbalanced, splitting the real observations before balancing also produces very small training and testing supports for the rarest attack classes. The 150,000-record subset contains 139,073 Normal observations, but only 32 CommInj and 27 Backdoor observations. Following the 80:20 split, approximately 22 Backdoor and 26 CommInj observations remain available for training, whereas the final test partition contains only 5 Backdoor and 6 CommInj observations. Thus, although the split-first protocol provides stronger protection against leakage, the statistical reliability of class-wise evaluation for these extremely rare attacks is necessarily limited by their very small real test support. Tables 22–24 report the class-wise results obtained under this leakage-controlled protocol.

Table 22. Class-wise performance of PH-BiLSTM-FCNN under the leakage-controlled split-first and training-only balancing protocol.

Class	Support	TP	FN	Precision	Recall	Specificity	F1-Score	ROC-AUC	PR-AUC
Backdoor	5	3	2	0.7500	0.6000	1.0000	0.6667	0.9985	0.6346
CommInj	6	6	0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
DoS	1967	1967	0	0.9990	1.0000	0.9999	0.9995	1.0000	1.0000
Reconn	207	207	0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Normal	27,815	27,814	1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Total	30,000	29,997	3	–	–	–	–	–	–

Table 23. Class-wise performance of Q-PH-BiLSTM-FCNN under the leakage-controlled split-first and training-only balancing protocol.

Class	Support	TP	FN	Precision	Recall	Specificity	F1-Score	ROC-AUC	PR-AUC
Backdoor	5	4	1	1.0000	0.8000	1.0000	0.8889	0.9980	0.8033
CommInj	6	6	0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
DoS	1967	1967	0	0.9995	1.0000	1.0000	0.9997	1.0000	0.9995
Reconn	207	207	0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Normal	27,815	27,815	0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Total	30,000	29,999	1	–	–	–	–	–	–

Table 22 shows that PH-BiLSTM-FCNN correctly classifies 29,997 of the 30,000 real test observations. The large Normal, DoS, and Recon classes achieve almost complete detection. However, the result for the Backdoor class is substantially more sensitive to individual errors because only five real Backdoor observations are available in the test partition. Three of the five Backdoor observations are correctly identified, resulting in a recall of 0.6000 and an F1-score of 0.6667. Although the corresponding ROC-AUC remains high at 0.9985, the PR-AUC decreases to 0.6346. This difference is important because PR-AUC is more sensitive to the precision-recall behavior of a rare class and therefore provides additional information that can be hidden by global accuracy or ROC-AUC.

Table 24. Class-wise performance of EAL-PH-BiLSTM-FCNN under the leakage-controlled split-first and training-only balancing protocol.

Class	Support	TP	FN	Precision	Recall	Specificity	F1-Score	ROC-AUC	PR-AUC
Backdoor	5	3	2	0.7500	0.6000	1.0000	0.6667	0.9974	0.5744
CommInj	6	6	0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
DoS	1967	1967	0	0.9990	1.0000	0.9999	0.9995	1.0000	0.9999
Reconn	207	207	0	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Normal	27,815	27,814	1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
Total	30,000	29,997	3	–	–	–	–	–	–

The Q-PH-BiLSTM-FCNN model provides the strongest result under the split-first evaluation, correctly classifying 29,999 of 30,000 observations. For the Backdoor class, four of five samples are detected, increasing recall from 0.6000 to 0.8000 and F1-score from 0.6667 to 0.8889 compared with PH-BiLSTM-FCNN. The Backdoor PR-AUC also increases from 0.6346 to 0.8033. Nevertheless, these values should still be interpreted cautiously because they are calculated from only five real Backdoor test observations.

A similar pattern is observed for EAL-PH-BiLSTM-FCNN. The model correctly classifies 29,997 of 30,000 test observations, while Normal, DoS, Recon, and CommInj are detected almost completely. However, only three of the five Backdoor observations are correctly classified. Consequently, Backdoor recall and F1-score decrease to 0.6000 and 0.6667, respectively, while PR-AUC reaches 0.5744. The results also demonstrate an important distinction between overall classification performance and statistical reliability of minority-class estimates. The PH-BiLSTM-FCNN and EAL-PH-BiLSTM-FCNN models make only three errors among 30,000 test observations, while Q-PH-BiLSTM-FCNN makes only one. Nevertheless, two Backdoor errors reduce recall to 0.6000 because the class contains only five test observations. Similarly, the CommInj class contains only six test observations; therefore, a single incorrect prediction would reduce its recall from 1.0000 to approximately 0.8333. Consequently, a recall of 1.0000 based on six observations should not be considered statistically equivalent to a recall of 1.0000 based on hundreds or thousands of independent observations. Previous sample-size studies have similarly shown that classification estimates obtained from very small numbers of independent test observations can have substantial uncertainty, particularly for sensitivity and other class-specific measures. The split-first protocol also introduces a limitation during model development. Because only approximately 22 Backdoor and 26 CommInj real observations remain in the training partition, each class-specific autoencoder must learn its minority-class latent distribution from a very limited number of genuine examples. Synthetic augmentation can increase the numerical number of training observations, but it cannot introduce the complete range of attack behavior that is absent from the original minority samples. Therefore, the quality and diversity of the generated minority observations remain constrained by the limited real examples available to the generator. This provides an additional reason for interpreting the leakage-controlled results together with the available class support rather than relying exclusively on aggregate classification metrics.

Accordingly, the two evaluation protocols are retained for different and complementary purposes. The primary Balancing → Splitting → Training setting is treated as a controlled class-balanced benchmark, in which each attack category has sufficient representation for stable model-wise and class-wise comparisons. Its results are not interpreted as a fully independent assessment of real-world minority-class generalization because synthetic observations are generated before the final partition. In contrast, the present split → training-only balancing → testing setting provides the stronger leakage-control experiment because the final test observations are completely isolated from preprocessing and synthetic generation. Therefore, neither protocol is used alone to support unrestricted

claims of real-world generalization. The balanced benchmark evaluates the comparative learning behavior of the proposed models when sufficient representation is available for all classes, whereas the split-first experiment evaluates whether high predictive performance remains when the final test observations are completely independent of synthetic generation. Reporting both settings makes the influence of the balancing protocol explicit and provides a more transparent interpretation of the experimental results.

4.14. Permutation Feature Importance Analysis

PFI is used to evaluate the relative contribution of input features by measuring the drop in classification accuracy after permuting each feature independently [25]. Features causing a larger accuracy reduction are considered more influential, making this approach suitable for interpreting the hybrid PH-BiLSTM-FCNN without altering its internal architecture.

Figure 10 presents the permutation importance results for the PH-BiLSTM-FCNN model. TotAppByte emerges as the most influential feature, producing the largest accuracy drop of approximately 0.32, followed by IdleTime with a reduction of around 0.25, indicating the critical role of application-level traffic volume and temporal inactivity patterns in intrusion detection. The feature sTtl also shows notable importance with an accuracy decrease above 0.15, reflecting the relevance of protocol-level behavior. A group of flow- and load-related features, including SrcBytes, SrcJitAct, TotBytes, and Load, exhibit moderate contributions with accuracy drops in the range of 0.12 to 0.14, capturing essential traffic intensity and source-side dynamics. In contrast, features such as Dport, DstLoad, and SrcPkts result in minimal accuracy degradation below 0.03, indicating limited influence on the final predictions.

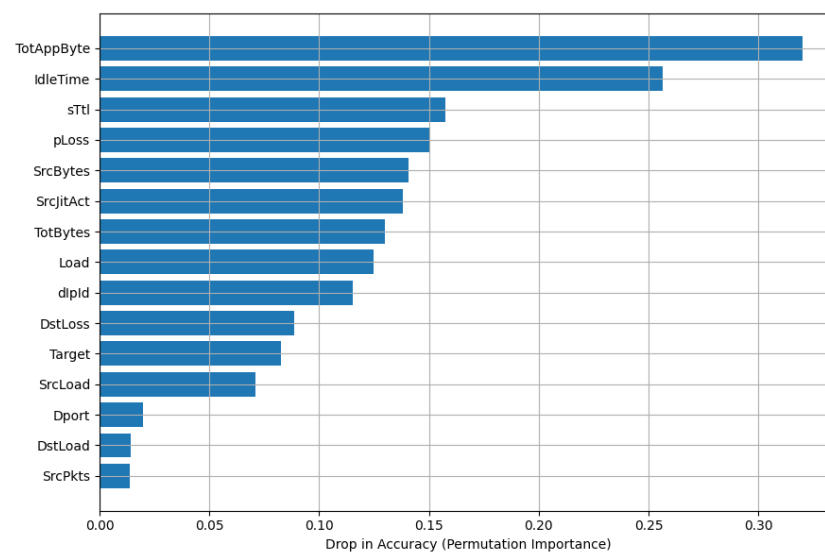


Figure 10. Permutation feature importance summary plot for PH-BiLSTM-FCNN.

Figure 11 presents the permutation importance results for the Q-PH-BiLSTM-FCNN model. IdleTime emerges as the most influential feature, producing the largest accuracy drop of approximately 0.41, highlighting the dominant role of traffic inactivity patterns in reinforcement-guided learning. This is followed by TotBytes, which causes an accuracy reduction of around 0.31, emphasizing the importance of overall traffic volume. The feature SrcBytes also shows a substantial contribution with an accuracy drop exceeding 0.22, reflecting the relevance of source-side data transmission behavior. A group of packet- and flow-level attributes, including TotPkts, sIpId, Sport, SrcLoad, DstPkts, and SrcPkts, exhibit moderate importance, with accuracy drops clustered around 0.16 to 0.18. These features collectively capture packet dispersion, source–destination interaction, and traffic

rate characteristics that are critical for detecting coordinated intrusion activities. In contrast, protocol-specific and destination-oriented features such as Dport, SrcRate, sTtl, TotAppByte, and DstBytes demonstrate comparatively lower importance, with accuracy reductions close to 0.15, indicating a supporting but non-dominant role in the Q-PH-BiLSTM-FCNN decision process.

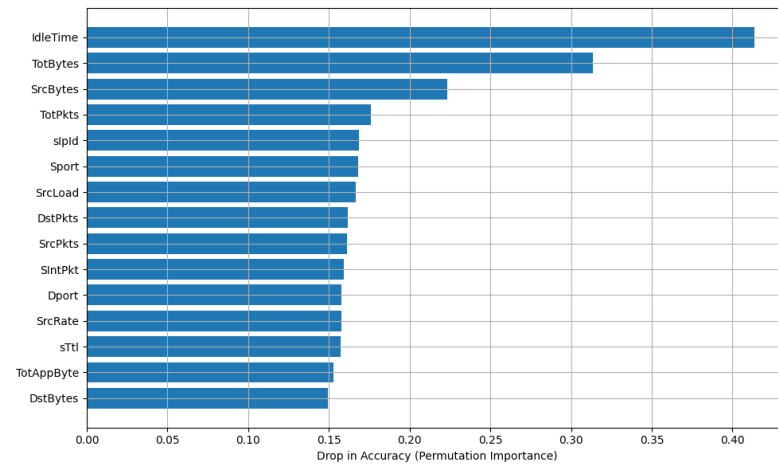


Figure 11. Permutation feature importance summary plot for Q-PH-BiLSTM-FCNN.

Figure 12 presents the PFI analysis for the proposed EAL-PH-BiLSTM-FCNN model. The results quantify the impact of each input feature by measuring the reduction in classification accuracy when feature values are randomly permuted. The features TotAppByte and TotBytes exhibit the highest accuracy degradation, indicating that aggregate application-level and flow-level byte statistics are the most influential factors in the model's decision-making process. Temporal and traffic behavior features such as IdleTime and SrcBytes also show substantial importance, reflecting the model's sensitivity to flow duration and source-side traffic intensity. Network load and destination-related attributes, including DstLoad and DstBytes, contribute moderately, suggesting their role in capturing bidirectional traffic patterns. In contrast, packet-level and rate-based features such as SrcPkts, Rate, and sIpId demonstrate comparatively lower importance, implying that their information content is partially redundant when higher-level traffic volume and temporal characteristics are available.

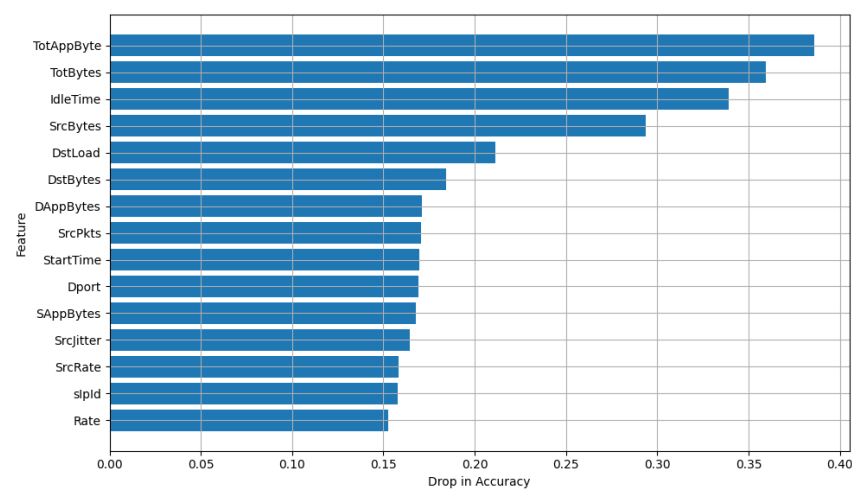


Figure 12. Permutation feature importance summary plot for EAL-PH-BiLSTM-FCNN.

5. Conclusions

This study introduced a detailed intrusion detection system that suits IoT settings, taking into consideration a few long-standing issues witnessed by the current learning-based security systems. The framework successfully alleviates the problem of class imbalance by integrating an autoencoder-based data balancing mechanism that allows the framework to provide more confident learning in both majority and minority classes of attack. The proposed PH-BiLSTM-FCNN takes advantage of parallel feature interaction and non-linear feature representation learning, leading to significant performance enhancement compared to traditional architectures. The Q-PH-BiLSTM-FCNN is applied to improve the convergence stability and detection accuracy enhancement by adding the adaptive training behavior. Moreover, the EAL-PH-BiLSTM-FCNN also shows that it can achieve good detection results and significantly minimize labeling dependency, which is more feasible in the real-world deployment of IoT. Extensive 10-FCV experiments demonstrate stable comparative performance of the proposed models across different partitions of the controlled balanced benchmark, while generalization to untouched real observations is assessed separately through the leakage-controlled split-first and external-dataset evaluations. Quantitatively, PH-BiLSTM-FCNN achieves an accuracy of 0.8451, an F1-score of 0.8441, and a ROC-AUC of 0.9654. Within the controlled balanced benchmark, Q-PH-BiLSTM-FCNN provides the strongest performance among the proposed variants, attaining an accuracy of 0.9154, an F1-score of 0.9144, a ROC-AUC of 0.9905, and reducing LL to 0.1489, highlighting its superior learning efficiency and stability. Meanwhile, EAL-PH-BiLSTM-FCNN achieves a competitive accuracy of 0.8616 and an F1-score of 0.8584 while significantly reducing ET to 324.28 seconds, demonstrating an effective trade-off between performance and computational efficiency. The Nadeau-Bengio dependence-corrected statistical analysis provides further evidence that the observed improvements remain consistent across the matched cross-validation folds, while the PFI analysis offers interpretable insights into the contribution of individual features. Finally, the use of blockchain and smart contracts in all three proposed models ensures the existence of transparent checks, unalterable performance logs and greater trustworthiness, making the proposed framework applicable for implementing secure and auditable deployments of IoT intrusion detection. The proposed framework further strengthens algorithm design by clearly defining each learning step and analyzing its performance, stability, and computational cost.

Limitations and Future Work

The proposed models show a high level of performance, but there are a number of limitations. First, the experimental assessment is performed mainly on one benchmark dataset that is gathered under controlled IIoT conditions. Despite the use of 10-FCV to enhance robustness, the learned representations and decision boundaries can still be sensitive to the statistical properties of this dataset, which undermines the validity of the proposed framework on other real-world IoT deployments with different traffic distributions, protocols, and unobservable attack patterns. Second, the autoencoder-based balancing scheme and RL optimization add additional computational load to the training, and these could restrict scalability or real-time practicality in very resource-constrained IoT edge settings. Although the dense representation layers and PFI enhance interpretability, the general architecture is still complicated, which complicates deployment, tuning, and maintenance of the system as opposed to lightweight IDS solutions.

The present blockchain evaluation is limited to a controlled permissioned deployment and primarily assesses auditability, record-integrity verification, transaction latency, throughput, and storage overhead. Under the current threat model, the blockchain layer protects logged IDS decisions, active-learning records, and optimization-related transac-

tions against unauthorized modification after commitment, while the integrity of the input traffic, correctness of the trained model, authenticity of initially authorized validators, and security of endpoints generating the records remain trusted assumptions. Therefore, the current experiments should not be interpreted as a comprehensive demonstration of resilience against Byzantine behavior, coordinated validator compromise, replay attacks, forged transactions, conflicting updates, or communication failures.

The current recurrent branch uses an input representation of shape $(1, d)$, where the complete feature vector forms one sequence element. Consequently, the dense representation layers cannot assign separate weights to individual predictor variables. The branch should therefore be interpreted as a single-step bidirectional nonlinear representation pathway rather than an explicit feature-selection mechanism. Although the revised leakage-controlled pipeline isolates the final test set and fits preprocessing only on training data, its near-perfect validation performance may still reflect overfitting, shortcut learning, or correlated and near-duplicate traffic samples across randomly generated partitions.

Future work will focus on extending the generalizability and practical applicability of the proposed framework across diverse and heterogeneous IoT and IIoT environments. In addition to the current cross-dataset validation, further experiments will investigate more stringent device-aware, session-aware, time-aware, chronological, group-disjoint, and spatiotemporal splitting strategies to reduce correlations among training and evaluation samples and provide a stronger assessment of generalization across different devices, sessions, and traffic-capture periods. These evaluations will also be complemented by unseen-attack analysis, false-alarm assessment, and model-calibration analysis. Domain adaptation and transfer-learning strategies will also be investigated to address distribution shifts between different networks, devices, and attack scenarios. Moreover, lightweight model compression, pruning, and quantization will be explored to reduce computational and memory requirements and enable deployment on representative edge hardware.

Future blockchain evaluation will incorporate explicit adversarial and fault-injection scenarios, including malicious or colluding validators, forged and replayed ledger records, individual and multiple validator failures, conflicting model or audit updates, delayed or dropped messages, and temporary network partitions. These experiments will be used to quantify Byzantine-fault tolerance, consensus robustness, record-rejection behavior, recovery characteristics, and the security-performance trade-off under increasingly hostile deployment conditions. Future extensions will further incorporate online and continual learning mechanisms for adapting to evolving attack behaviors in real time, as well as advanced generative balancing methods that preserve temporal, protocol-level, and class-specific traffic characteristics. Feature-wise tokenization will also be investigated using an input representation of shape $(d, 1)$, in which each predictor variable is treated as an independent sequence element. Finally, additional external datasets, duplicate-flow analysis, and more extensive synthetic-sample diversity and nearest-neighbor similarity assessments will be considered to provide stronger evidence of robustness and generalization.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Data Availability Statement: The datasets used in this study are publicly accessible and can be found at: <https://www.kaggle.com/datasets/annaamalaiu/wustl-iiot-2021-dataset>, accessed on 28 January 2026; <https://www.kaggle.com/datasets/subhajournal/iotintrusion>, accessed on 31 July 2026.

Conflicts of Interest: The author declares no conflicts of interest.

References

1. Ashfaq, M.; Nur, S. IoT Sensor Networks-Orchestrating Connectivity, Efficiency, and Intelligence Across Diverse Domains. *Int. J. Innov. Res. Comput. Sci. Technol.* **2024**, *12*, 154–161. [\[CrossRef\]](#)
2. Javed, A.; Javaid, N.; Awan, K.M.; Ahmed, I.; Pamucar, D.; Shafiq, M.; Choi, J.G. ReGAIN: A reinforcement-enhanced generative AI framework for intelligent intrusion detection in IoT networks. *Complex Intell. Syst.* **2026**, *12*, 127. [\[CrossRef\]](#)
3. Alani, M.M.; Awad, A.I. An intelligent two-layer intrusion detection system for the Internet of Things. *IEEE Trans. Ind. Inform.* **2022**, *19*, 683–692. [\[CrossRef\]](#)
4. Sebestyen, H.; Popescu, D.E.; Zmaranda, R.D. A literature review on security in the Internet of Things: Identifying and analysing critical categories. *Computers* **2025**, *14*, 61. [\[CrossRef\]](#)
5. Babak, V.; Babak, S.; Eremenko, V.; Kuts, Y.; Zaporozhets, A. Protection of Measurement Information from Unauthorized Access. In *Information-Measuring Systems: Theory and Application*; Springer Nature: Cham, Switzerland, 2025; pp. 409–458. [\[CrossRef\]](#)
6. Thapliyal, S.; Wazid, M.; Singh, D.P. Design of Distributed Denial-of-Service Attack Mechanism for IoT-Driven Data Fusion System. *Cyber Secur. Appl.* **2025**, *3*, 100092. [\[CrossRef\]](#)
7. Saura, J.R.; Ribeiro-Soriano, D.; Palacios-Marqués, D. Setting privacy “by default” in social IoT: Theorizing the challenges and directions in Big Data Research. *Big Data Res.* **2021**, *25*, 100245. [\[CrossRef\]](#)
8. Anwer, R.W.; Abrar, M.; Ullah, M.; Salam, A.; Ullah, F. Advanced intrusion detection in the industrial Internet of Things using federated learning and LSTM models. *Ad Hoc Netw.* **2025**, *178*, 103991. [\[CrossRef\]](#)
9. Adefemi, K.O.; Mutanga, M.B.; Alimi, O.A. A Hybrid CNN–GRU Deep Learning Model for IoT Network Intrusion Detection. *J. Sens. Actuator Netw.* **2025**, *14*, 96. [\[CrossRef\]](#)
10. Chalichalamala, S.; Govindan, N.; Kasarapu, R. Logistic regression ensemble classifier for intrusion detection system in internet of things. *Sensors* **2023**, *23*, 9583. [\[CrossRef\]](#) [\[PubMed\]](#)
11. Guerra-Manzanares, A.; Bahsi, H. On the application of active learning for efficient and effective IoT botnet detection. *Future Gener. Comput. Syst.* **2023**, *141*, 40–53. [\[CrossRef\]](#)
12. Das, S.; Majumder, A.; Namasudra, S.; Singh, A. Intrusion detection using CTGAN and lightweight neural network for Internet of Things. *Expert Syst.* **2025**, *42*, e13793. [\[CrossRef\]](#)
13. Eid, A.M.; Soudan, B.; Nassif, A.B.; Injadat, M. Enhancing intrusion detection in IIoT: Optimized CNN model with multi-class SMOTE balancing. *Neural Comput. Appl.* **2024**, *36*, 14643–14659. [\[CrossRef\]](#)
14. Shaikh, J.A.; Wang, C.; Sima, M.W.U.; Arshad, M.; Owais, M.; Hassan, D.S.; Alkanhel, R.; Muthanna, M.S.A. A deep Reinforcement learning-based robust Intrusion Detection System for securing IoMT Healthcare Networks. *Front. Med.* **2025**, *12*, 1524286. [\[CrossRef\]](#) [\[PubMed\]](#)
15. Ch, M.M.S.S.; Rao, Y.N. Design of an iterative method leveraging deep Q-networks for intrusion detection system operations. *IEEE Access* **2025**, *13*, 48720–48745. [\[CrossRef\]](#)
16. Salehiyan, A.; Moghaddam, P.S.; Kaveh, M. An Optimized Transformer–GAN–AE for Intrusion Detection in Edge and IIoT Systems: Experimental Insights from WUSTL-IIoT-2021, EdgeIIoTset, and TON_IoT Datasets. *Future Internet* **2025**, *17*, 279. [\[CrossRef\]](#)
17. Anitha, M.; Savarimuthu, N.; Bhanu, S. Chi-square target encoding for categorical data representation: A real-world sensor data case study. *SN Comput. Sci.* **2025**, *6*, 228. [\[CrossRef\]](#)
18. Hasnain, M.; Javaid, N.; Saudagar, A.K.J.; Kumar, N. An intelligent and explainable intrusion detection framework for Internet of Sensor Things using generalizable optimized active Machine Learning. *J. Netw. Comput. Appl.* **2025**, *245*, 104358. [\[CrossRef\]](#)
19. Zhang, C.; Bom, S. Auto-encoder based model for high-dimensional imbalanced industrial data. In *Proceedings of the International Conference on Neural Information Processing, Virtual Event, 8–12 December 2021*; Springer International Publishing: Cham, Switzerland, 2021; pp. 265–273. [\[CrossRef\]](#)
20. Alavizadeh, H.; Alavizadeh, H.; Jang-Jaccard, J. Deep Q-learning based reinforcement learning approach for network intrusion detection. *Computers* **2022**, *11*, 41. [\[CrossRef\]](#)
21. Chen, D.W.; Jin, Y.H. An active learning algorithm based on shannon entropy for constraint-based clustering. *IEEE Access* **2020**, *8*, 171447–171456. [\[CrossRef\]](#)
22. Alimbayev, C.; Alimbayeva, Z.; Ozhikenov, K.; Karibayev, K.; Abuova, Z.; Akhmedova, D. Interpretable Machine Learning Approach for Diabetes Classification in Patients with Cardiovascular Disease. *Algorithms* **2026**, *19*, 546. [\[CrossRef\]](#)
23. Subhajournal. IoT Intrusion Detection, Kaggle. Available online: <https://www.kaggle.com/datasets/subhajournal/iotintrusion> (accessed on 31 July 2026).

24. Sabir, L.; Javaid, N.; Akbar, M.; Alrajeh, N.; Hussain Bouk, S.; Aldegheishem, A. DeepSecure: A novel deep learning model for effective detection of attacks on big data in Internet of Urban Things. *J. Big Data* **2025**, *12*, 273. [[CrossRef](#)]
25. Antoska Knights, V.; Mazlami, V.; Prchkovska, M.; Gajdoš Kljusurić, J. A unified interpretability framework for feature importance in machine learning models. *Algorithms* **2026**, *19*, 548. [[CrossRef](#)]

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.