Multi-task Temporal Deep Learning Model for Real Time Intrusion Detection System

Christian Budhi Sabdana [1,*)], Noriandini Dewi Salyasari [2)], Izra Noor Zahara Aliya [3)], and Ary Mazharuddin Shiddiqi [4)]

[1)] SAP Division, PT 360 Teknologi Indonesia
Jakarta, Indonesia

[2)] Information Technology and Information Systems Division, RSUD dr Soedono Provinsi Jawa Timur
Madiun, Indonesia

[3)] Finance System Division, PT Social Bella indonesia
Jakarta, Indonesia

[4)] Department of Informatics, Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia

E-mail: cbsabdana@gmail.com[1)], salyasari@gmail.com[2)], izra.aliya28@gmail.com[3)], and ary.shiddiqi@its.ac.id[4)]

ABSTRACT

The rapid expansion of Internet of Things (IoT) ecosystems has enabled large-scale interconnected smart environments while simultaneously exposing IoT devices to increasingly sophisticated cyber threats. To address these challenges, machine learning and deep learning based intrusion detection systems (IDS) have been widely adopted; however, many existing approaches suffer from insufficient temporal modeling, and poor performance under extreme class imbalance. In this study, we investigate a multi-task stacked Long Short-Term Memory (LSTM) architecture for IoT intrusion detection, where binary anomaly detection and multi-class attack classification are jointly learned within a unified temporal framework. The proposed model examines different inter-path knowledge transfer mechanisms, including additive, gated, and attention-based aggregation, to enhance discriminative attack representation learning. A topology-constrained shuffling strategy is further introduced to preserve intra-flow temporal dependencies while reducing reliance on fixed traffic ordering. Experimental results on the Edge-IIoTset dataset show that all models achieve high binary detection performance (F1-score above 97%), while attention-based aggregation consistently outperforms static fusion strategies for multi-class classification, yielding superior macro F1-score and AUC-PR under severe class imbalance. These findings emphasize the importance of context-aware information sharing and temporal structure preservation for robust and adaptive IoT intrusion detection systems.

Keywords: Intrusion detection system, temporal deep learning, multi-task prediction, long-short term memory.

1. Introduction

The rapid growth of the Internet of Things (IoT) ecosystem has led to the deployment of large-scale interconnected intelligent devices, ranging from industrial systems to household sensors [1]. The Internet of Things represents a new phase in the evolution of the internet, in which physical objects are equipped with sensing capabilities and the ability to communicate with one another. This technology has been widely adopted across various industrial sectors [2].

However, numerous studies [3], [4], [5] have shown that IoT devices exhibit a high level of vulnerability due to the heterogeneity of their networks, limited security capabilities, and massive deployment scale. These conditions enable attackers to compromise millions of IoT devices and incorporate them into botnets, which are then leveraged to launch large-scale distributed denial-of-service (DDoS) attacks that overwhelm networks and disrupt

* Corresponding author.
Received: December 22nd, 2025. Revised: December 26th, 2025. Accepted: January 4th, 2026.
Available online: January 15th, 2026.

DOI: https://doi.org/10.12962/j24068535.v24i1.a1446

server operations [6]. Consequently, there is a pressing need for intrusion detection approaches that go beyond static data characteristics and are capable of capturing the temporal dynamics of network traffic while adapting to heterogeneous operational environments [7].

Machine Learning (ML) and Deep Learning (DL)-based intrusion detection methods have been increasingly adopted in IoT environments, as they demonstrate higher detection accuracy than traditional rule-based techniques [1], [8]. Prior research indicates that algorithms such as Support Vector Machines (SVM), Random Forests, Neural Networks, Long Short-Term Memory (LSTM) networks, and Convolutional Neural Networks (CNN) are effective in enhancing IoT security [6], [9], [10]. Nevertheless, many of these approaches still suffer from significant generalization limitations when deployed on real-time data [11]. A considerable number of studies design their models to operate on aggregated flow-level features or apply recurrent architectures to packets that have already been grouped into flows [12]. While these techniques perform well under experimental conditions, they introduce substantial computational overhead when flow reconstruction must be performed in real time [12], [13]. Moreover, such models often achieve optimal performance only on the datasets used during training, while their effectiveness degrades markedly when applied to different domains or previously unseen attack scenarios [14]. This limitation is primarily attributed to reliance on device-specific features, training processes that insufficiently capture the temporal structure of traffic flows, and a lack of rigorous cross-dataset evaluation that reflects real-world variability [11], [13]. Therefore, more adaptive modeling approaches capable of abstracting features at a higher and more generic level are critically needed for modern IoT security systems.

Motivated by these challenges, this study proposes an IoT intrusion detection model based on a Multi-Task Stacked LSTM architecture combined with an adaptive feature engineering strategy to enhance cross-domain attack pattern understanding. The Multi-Task Stacked LSTM integrates multi-task learning with stacked LSTM layers, enabling the model to learn complex temporal patterns while simultaneously addressing multiple related tasks. This approach improves both learning efficiency and generalization capability and has been successfully applied to various time-series problems [15], [16], [17]. Rather than relying solely on static representations, the proposed method exploits sequential traffic information without performing flow-level grouping or aggregation. Feature selection is guided by domain knowledge to emphasize device-agnostic characteristics and avoid overfitting to specific physical device attributes. In addition, a gated aggregation mechanism is employed during node representation fusion to highlight the most relevant features for classification.

The contributions of this research are twofold. First, this study proposes a Multi-Task Stacked LSTM-based intrusion detection framework that jointly models temporal dependencies in IoT network traffic while learning multiple related detection objectives, enabling more effective representation learning for sequential traffic patterns. Second, an adaptive feature engineering strategy guided by domain knowledge is introduced to prioritize device-agnostic and flow-relevant features, thereby reducing dependence on hardware-specific attributes. This strategy is further supported by a shuffling mechanism, which preserves temporal ordering within traffic flows without requiring expensive real-time flow reconstruction.

The rest of this article is organized as follows. Section 2 presents multiple related research on Intrusion Detection System using Deep Learning algorithm. Section 3 describes the research methodology, including the experimental design for attack prediction and temporal modelling to train the models. Section 4 reports the experimental results with comparison between multiple architecture design. Section 5 discusses the key findings, analysis, implication of our experimental results, and the limitation of this works. Section 6 concludes the research and provides directions for further study.

2. Related Works

This section reviews prior studies relevant to this research, as summarized in Table 1. The analysis aims to provide a comprehensive understanding of the evolution of intrusion detection system (IDS) research and to position the present study within the existing body of work. Over the years, IDS methodologies have evolved from classical offline evaluations using benchmark datasets to more dynamic, real time detection settings. In particular, recent

Table 1: Previous Related Works in Intrusion Detection Systems using Deep Learning.

Year/Ref.	Method	Temporal	Imbalance Data Handling	Target	Multitask	Dataset	Result
Research employing Classical IDS Benchmark Datasets							
2017 [18]	Recurrent Neural Network (RNN-IDS)	Yes	No	Binary Multiclass	No	KDD99	Accuracy 97.5% Precision 96%
2018 [19]	Stacked Non-symmetric Deep Autoencoder + Random Forest (NDAE+RF)	No	No	Binary Multiclass	No	KDD99, NSL-KDD	Accuracy 97.9% F1-score 97%
2019 [20]	Deep Neural Network (Scale-Hybrid-IDS-AlertNet)	No	No	Binary Multiclass	Yes	KDDCup99, NSL-KDD, UNSW-NB15, Kyoto, CICIDS2017	Accuracy 99.2% F1-score 98.7%
2022 [21]	Deep Neural Network (DNN) + Multi-Task Learning (MTL)	No	No	Binary	Yes	UNSW-NB15, CICIDS2017	Improved performance (not numerically stated)
2023 [22]	DNN + filter-based feature selection + GAN synthetic data	No	Generative Adversarial Network (GAN)	Binary Multiclass	No	UNSW-NB15	Accuracy 84% (without GAN) Accuracy 91% (with GAN)
2023 [23]	CNN + CapSA (Hybrid evaluation)	No	No	Binary Multiclass	No	NSL-KDD, BoT-IoT, KDD99, CIC2017	Accuracy 99.1% Precision 98.9% Recall 98.7%
Research employing Real-time and IoT-oriented IDS Datasets							
2023 [24]	Stacking ensemble (CNN + LSTM + GRU + DNN)	Yes	No Stratified KFold to preserve labels ratio	Binary Multiclass	No	ToN_IoT, CICIDS2017, SWaT	Accuracy 99.4% FPR < 1%
2023 [25]	Ensemble of RNNs (LSTM + GRU) + Harris Hawk Optimization	Yes	No	Multiclass	No	IoT benchmark datasets	Accuracy 98.5% Precision 98% Recall 97.8%
2023 [26]	FFNN, LSTM, Random Neural Network (RandNN)	Yes	Synthetic Minority Oversampling (SMOTE)	Binary	No	CIC-IoT22	F1-Score: FFNN 99.93% LSTM 99.85% RandNN 96.42%
2024 [27]	CNN-GRU hybrid (AttackNet)	Yes	No	Multiclass	No	N_BaIoT	Accuracy 99.8% Precision 99.8% Recall 99.7%
2025 [28]	Self-supervised Contrastive Learning (CARLA)	Yes	No	Binary	No	7 TSAD real-world datasets	F1-score 97.2% AU-PR 98%

studies have explored diverse frameworks such as Deep Learning (DL) and Multi-Task Learning (MTL) to enhance detection performance across both traditional and IoT oriented environments. These developments form the basis for the comparative overview presented in the next section.

Table 1 summarizes deep learning based IDS studies, categorized by dataset type (classical and real-time IoT). This grouping illustrates the gradual evolution from traditional offline evaluation to modern IoT based and

streaming detection. Since these studies differ in their goals and evaluation settings, the performance results are reported as presented in the original papers. They should be interpreted alongside the “Imbalance Data Handling” column, as all benchmark and IoT datasets are naturally imbalanced and were managed using the methods listed in that column.

Based on the datasets reviewed in Table 1, several classical benchmarks such as KDD99 [29], NSL-KDD [29], and UNSW-NB15 [30] were developed under controlled or synthetic conditions that no longer reflect the complexity of contemporary network traffic. Prior studies have critically noted that these datasets oversimplify attack behaviors and fail to represent the dynamic and heterogeneous characteristics of real world IoT environments [31], [32]. Consequently, models trained on such data often suffer from overfitting and limited generalization. While these datasets remain valuable for reproducibility and benchmarking, their restricted realism constrains the meaningful evaluation of modern deep learning based IDS models. In response, more recent datasets have shifted toward IoT oriented and temporally dynamic traffic, underscoring the growing need for IDS approaches capable of learning from imperfect, evolving, and heterogeneous data.

Several recent studies have explored deep learning and multitask learning approaches for intrusion detection in IoT environments, each addressing different challenges with varying degrees of success. Sanju et al. [23] proposed the MM-WMVEDL model, a hybrid intrusion detection framework that combines metaheuristic optimization with an ensemble of LSTM and GRU networks to handle the physical and functional heterogeneity of IoT systems, achieving an accuracy of 98.12%. However, this approach relied on a computationally intensive single task architecture and did not incorporate explicit mechanisms for class imbalance mitigation or zero day attack evaluation, limiting its adaptability to unseen traffic conditions.

In contrast, Albelwi et al. [19] introduced a Multi-Task Deep Learning (MTDL) framework integrating contrastive learning and supervised clustering to enhance feature representation and improve cross-dataset generalization. The model achieved accuracies ranging from 95.5% to 99.9% across NSL-KDD, AWID, and BoT-IoT datasets. Despite these promising results, the framework primarily emphasized spatial feature extraction and lacked temporal modeling an essential component for capturing the dynamics of IoT traffic.

Elsayed et al. [24] developed SATIDS, a two level LSTM based multitask architecture designed to classify both attack categories and subtypes, achieving 96.35% accuracy in IoT environments. However, the system was evaluated using closed set testing on a single dataset, without cross dataset or unseen attack validation, thereby limiting its ability to detect zero day intrusions. Although the multitask structure reduced overfitting, it continued to face severe class imbalance and insufficient generalization to novel attack patterns.

Overall, studies such as MM-WMVEDL, MTDL, and SATIDS have made significant contributions to improving intrusion detection performance in IoT environments. Nevertheless, several limitations remain. A comparative analysis of these studies reveals three major challenges. First, many IoT-oriented IDS models still depend on outdated or homogeneous datasets, which limits their ability to generalize across diverse and evolving network conditions. Second, although multitask learning improves generalization across attack categories, most architectures continue to process tasks independently, with minimal aggregation or feature fusion mechanisms between learning branches. The absence of explicit fusion strategies such as gated, additive, or attention based integration reduces these models’ capacity to share complementary representations and jointly optimize across related tasks. Third, the temporal characteristics of IoT traffic are often underutilized; random batching or non sequential training disrupts flow level continuity and weakens the model’s capacity to capture long range dependencies and temporal correlations. Collectively, these shortcomings highlight the need for IDS frameworks that integrate temporal modeling, inter task collaboration, and effective fusion driven learning while maintaining robustness to data imbalance directions that form the conceptual foundation of the present study. In particular, the proposed approach extends beyond existing fusion paradigms by introducing structured additive, gated, and cross attention mechanisms that enable dynamic and interpretable information exchange between tasks, addressing limitations observed in prior multitask IDS architectures.

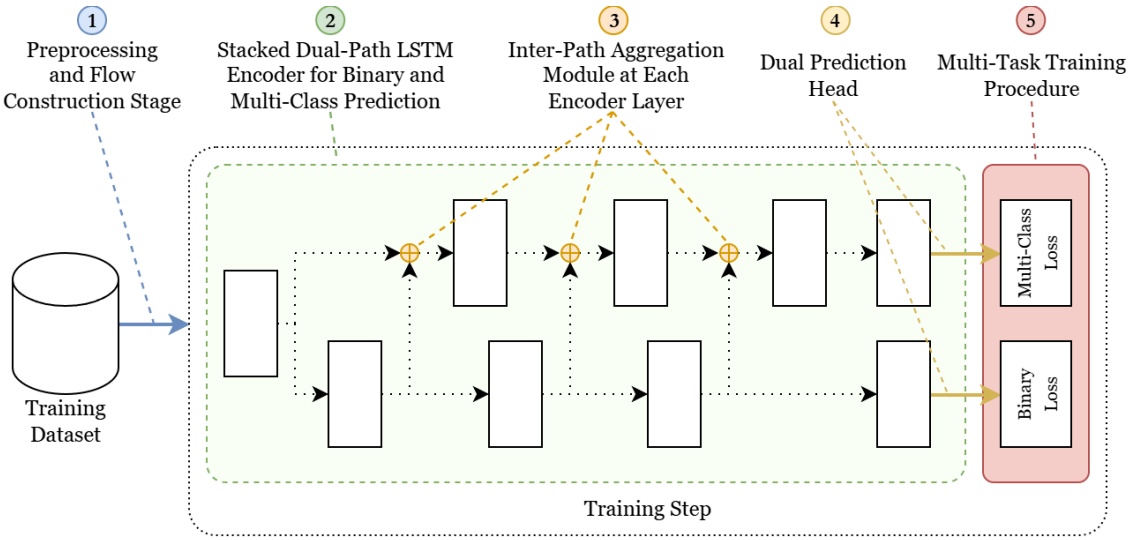

Fig. 1: Overview of the five main components. The Stacked Dual-Path LSTM (Component 2) is described in detail in Fig. 2, while the full training procedure is presented in Fig. 3.

To address these challenges, this study proposes a Stacked Dual-Path LSTM Encoder that integrates three aggregation mechanisms Additive, Gated, and Cross Attention to facilitate effective information exchange between anomaly detection and attack type classification tasks. Unlike conventional fusion strategies such as feature concatenation or shared layer multitask learning, the proposed architecture establishes structured communication pathways that explicitly regulate inter task interactions and feature alignment, enhancing cooperative representation learning. This design offers a more flexible and interpretable alternative to existing fusion schemes by allowing dynamic information exchange between tasks, instead of depending on static feature combination or limited parameter sharing. To further strengthen model robustness, the design incorporates imbalance aware optimization through a weighted focal loss, along with a structured shuffling mechanism that preserves flow level temporal topology during training, enabling more realistic modeling of IoT traffic. Additionally, this study investigates the influence of encoder depth, comparing three and four layer configurations to evaluate their effects on the richness and stability of temporal representations. Overall, the proposed framework establishes a comprehensive and resilient foundation for IoT intrusion detection, effectively overcoming the architectural and data centric limitations that remain unaddressed in prior studies.

3. Proposed Method

The proposed model is a multi-task temporal deep learning architecture designed to accurately detect attacks in IoT networks while simultaneously identifying their attack types. The system processes flow-based packet sequences in order to preserve temporal dependencies, which are a critical characteristic of IoT communication traffic patterns. An overview of this system are shown in Fig. 1.

Overall, the architecture consists of five main components, described as follows:

1. Preprocessing and Flow Construction Stage
   This stage transforms raw network packets into structured sequential representations based on network flows, enabling consistent temporal modeling of packet-level information.
2. Stacked Dual-Path LSTM Encoder for Binary and Multi-Class Prediction
   The encoder learns temporal patterns from both normal and malicious traffic by modeling sequential dependencies across packet flows.
3. Inter-Path Aggregation Module at Each Encoder Layer
   This module facilitates knowledge sharing from the binary anomaly detection task to the multi-class attack classification task, improving representational consistency across tasks.

4. Dual Prediction Head
   The model produces two outputs simultaneously: anomaly detection (binary classification) and attack type identification (multi-class classification).
5. Multi-Task Training Procedure
   Model training employs a combination of Focal Loss and gradient-based optimization to effectively address class imbalance and stabilize multi-task learning.

The multi-task learning strategy hypothetically enables generalization to previously unseen (zero-day) attacks while increasing training efficiency by jointly learning both tasks within a single integrated model.

3.1. Preprocessing and Flow Construction Stage

After data cleaning and feature selection, the dataset is reorganized based on *(flow_id, flow_seq)* pairs. Each flow represents a sequence of packets originating from the same network connection, ordered by increasing timestamps. The flow construction in this study utilizes heuristics optimized for TCP and UDP traffic, which constitute the vast majority of the network communications in the benchmark datasets. This approach ensures reliable flow aggregation for connection-oriented protocols. A detailed discussion regarding the applicability of these heuristics to other protocol types is provided in section 5.

The flow construction in this study follows protocol-specific heuristics primarily designed for TCP and UDP traffic, which dominate the benchmark datasets used. While this approach provides reliable flow aggregation for connection-oriented communication, its direct applicability to non-TCP/UDP traffic (e.g., ICMP or CoAP) is limited and remains a direction for future enhancement. Each packet within a flow contains three main components:

1. Flow identifiers, consisting of *flow_id* and *flow_seq*, which indicate the flow identity and the packet order within the flow;
2. Selected and encoded numerical features; and
3. Attack class labels, where 0 denotes normal traffic and 1-14 correspond to different attack types.

Before the training data are fed into the model, a shuffling mechanism is applied that preserves the packet order (topology) within each flow while allowing reordering across different flows. Furthermore, this mechanism is referred to as topology-constrained shuffling. Mathematically, this mechanism can be formulated as sampling a global permutation σ from a restricted permutation space that preserves intra-flow temporal ordering while permitting arbitrary inter-flow rearrangement.

Let $S_i = \left(s_{i,1}, s_{i,2}, s_{i,3}, ...\right)$ denote the packet sequence of flow-i in the dataset, where each packet $s_{i,j}$ represents the j-th packet of flow sequence S_i. The global permutation σ is constrained to satisfy $s_{i,j} \prec s_{i,k} \Rightarrow \sigma\left(s_{i,j}\right) < \sigma\left(s_{i,k}\right)$, which ensures that the temporal causality of packets belonging to the same flow is strictly preserved. Here, $\sigma(s)$ denotes the position of packet s in the permuted sequence induced by σ.

Formally, the topology-constrained shuffling operation could be defined as:

$$H = \left(\sigma \in S \mid s_{i,j} \prec s_{i,k} \Rightarrow \sigma\left(s_{i,j}\right) < \sigma\left(s_{i,k}\right)\right) \tag{1}$$

In equation (1), H represents the resulting shuffled sequence (denoted using parentheses) obtained by applying the constrained permutation σ to the flattened flow sequence S.

This strategy enables the model to learn valid temporal patterns within individual flows while mitigating unintended dependencies on static network topology or fixed inter-flow ordering. The resulting structured sequence tensors serve as input to the temporal feature extraction stage of the Stacked LSTM Encoder.

3.2. Stacked Dual-Path LSTM Encoder for Binary and Multi-Class Prediction

The temporal feature extraction stage is implemented using multiple layers of a Stacked Dual-Path LSTM Encoder, as illustrated in Fig. 2. Each traffic flow is processed sequentially to capture both short-term and long-term temporal dependencies inherent in IoT network traffic. This design explicitly models the sequential nature of

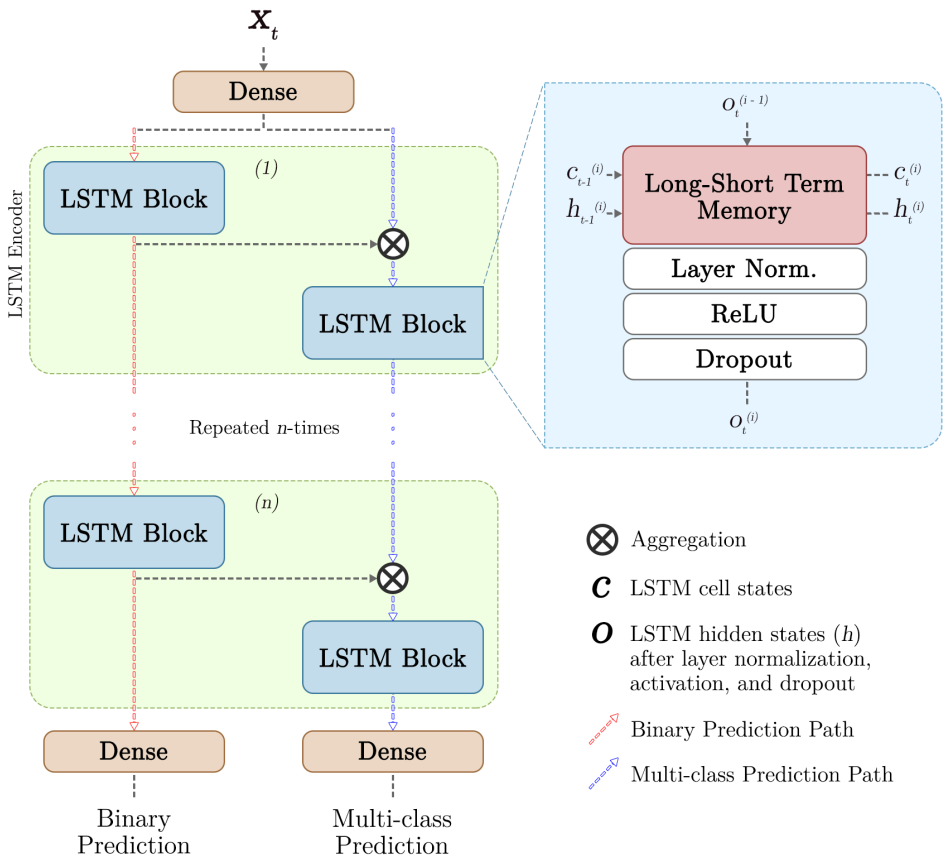

Fig. 2: Stacked Dual-Path LSTM Encoder; Deep Learning Architecture.

packet-level flows, which is essential for accurately characterizing normal communication behavior as well as the temporal dynamics of network attacks.

At each encoder layer, two parallel LSTM blocks are employed to serve distinct yet complementary learning objectives. Specifically:

1. *Binary Path*, which focuses on discriminating between normal and malicious traffic, and
2. *Multi-Class Path*, which aims to learn fine-grained temporal patterns associated with different attack categories.

Both paths share the same recurrent formulation but operate on task-specific representations. The computations performed within each path are defined by the standard LSTM recurrence and subsequent post-processing operations. Let $x_t \in \mathbb{R}^d$ represents input vectors at position-t in the sequence with dimension-d, each LSTM layer could formally defined as:

$$\begin{aligned} \left(h_t^{(i)}, c_t^{(i)}\right) &= \text{LSTM}\left(h_{t-1}^{(i)}, c_{t-1}^{(i)}, o_t^{(i-1)}\right) \\ o_t^{(i)} &= \text{Dropout}\left(\text{ReLU}\left(\text{LayerNorm}\left(h_t^{(i)}\right)\right)\right) \end{aligned} \tag{2}$$

From equation (2), $h_t^{(i)}$ and $c_t^{(i)}$ each correspond to hidden-state and cell-state output of LSTM block for position-t in the sequence and layers-(i) (see Fig. 2). While $o_t^{(i)}$ denote LSTM output (hidden-state $h_t^{(i)}$) followed by layer normalization, nonlinearity function ReLU, and dropout.

The post-processing stage serves multiple purposes: layer normalization stabilizes hidden-state distributions and mitigates exploding gradients, the ReLU activation alleviates vanishing gradient issues by promoting sparse and non-linear representations, and dropout reduces the risk of overfitting by introducing stochastic regularization. Although applying ReLU after LSTM is less common in conventional sequence models, in this architecture we

hypothesize ReLU could helps maintain stable gradient flow and enhances non-linear feature expressiveness across stacked recurrent layers. This design follows the normalization principles introduced in Layer Normalization [33], which have been shown to improve training stability and gradient propagation in deep sequential models. The necessity of this component was empirically validated through ablation studies, which are detailed in Section 4.4.

3.3. Inter-Path Aggregation Module at Each Encoder Layer

Before the input features are fed into the Multi-Class LSTM block, an inter-path aggregation process is applied using the feature representations produced by the Binary LSTM block. The primary objective of this mechanism is to facilitate knowledge transfer from the simpler task (binary anomaly detection) to the more complex task of multi-class attack classification. This asymmetric flow is intended to maintain the distinct functional roles of the dual-path architecture. By injecting coarse-grained, anomaly-aware information into the multi-class pathway, the model is guided to learn temporal representations that are more informative and discriminative for distinguishing among attack types with similar traffic patterns.

The aggregation function in equation (3), (4) and (6), denoted as $Agg(.,.)$, combines the feature representations $o_{\text{multi}}^{(i-1)}$ and $o_{\text{bin}}^{(i)}$ which correspond to the output of the $(i-1)$-th layer of the multi-class LSTM path and the output of the (i)-th layer of the binary LSTM path, respectively. This aggregation function is configurable and varies across experimental settings. In this study, three aggregation strategies are investigated:

1. Additive Aggregation

$$Agg\left(o_{\text{multi}}^{(i-1)}, o_{\text{bin}}^{(i)}\right) = o_{\text{multi}}^{(i-1)} + o_{\text{bin}}^{(i)} \tag{3}$$

   Aggregation function defined in equation (3) directly sums the representations from both paths without applying any feature selection or weighting mechanism.

2. Gated Aggregation

$$\begin{aligned} Agg\left(o_{\text{multi}}^{(i-1)}, o_{\text{bin}}^{(i)}\right) &= o_{\text{multi}}^{(i-1)}(1-\tanh(g)) + o_{\text{bin}}^{(i)}\tanh(g) \\ g &= W_g\left(o_{\text{multi}}^{(i-1)} \oplus o_{\text{bin}}^{(i)}\right) \end{aligned} \tag{4}$$

   In aggregation function defined with equation (4), g is the gating vector from transforming the concatenated ($\oplus$) outputs with learnable parameters W_g. The hyperbolic tangent activation is defined on equation (5) with e defined as euler constant:

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{5}$$

   The tanh function defined in equation (5) is chosen because LSTM outputs after ReLU lie in $[0, +\infty)$, where sigmoid quickly saturates and limits gating flexibility into $\left[\frac{1}{2}, 1\right]$. In contrast, tanh provides a smoother nonlinearity with a wider effective dynamic range, and can be linearly shifted to $[0, 1]$ for gating. This improves gradient flow and enables more expressive modulation of inter-path feature contributions.

3. Cross-Attention Aggregation

$$Agg\left(o_{\text{multi}}^{(i-1)}, o_{\text{bin}}^{(i)}\right) = \text{Attention}\left(o_{\text{multi}}^{(i-1)}, o_{\text{bin}}^{(i)}, o_{\text{bin}}^{(i)}\right) \tag{6}$$

   This aggregation function defined in equation (6) use attention mechanism that defined in equation (7), as:

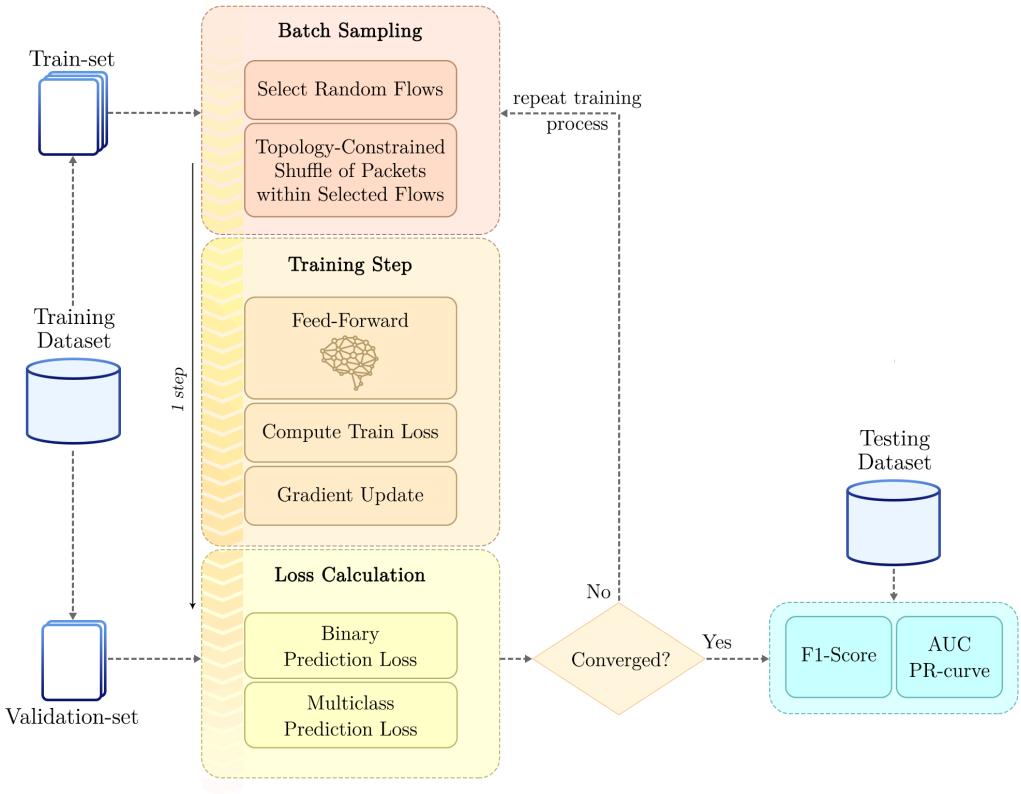

Fig. 3: Diagram of training process. Training is performed iteratively until model convergence, defined as no improvement in validation loss over a predefined number of iterations.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$
$$Q = W_q o_{\text{multi}}^{(i-1)} \tag{7}$$
$$K = W_k o_{\text{bin}}^{(i)}$$
$$V = W_v o_{\text{bin}}^{(i)}$$

Equation (7) is a standard scaled dot-product attention introduced by Vaswani et al [34]. In our setting, attention aggregation used to perform cross-attention between binary and multi-class output with W_q, W_k, and W_v each a learnable parameter, and softmax is activation function defined as:

$$\text{softmax}(x) = \frac{e^{x_i}}{\sum_{j=1}^{K} x_j} \tag{8}$$

In equation (8), x_i corresponds to each element in input vector x, and K as the dimension of input vector x.

This formulation allows the multi-class representation to selectively attend to binary-path features based on contextual relevance, enabling fine-grained feature alignment across tasks.

After all encoder layers are processed, the final temporal representation is used to generate two model outputs. The Binary Classification Head distinguishes normal traffic from attack traffic, while the Multi-Class Classification Head identifies specific attack types. Each output is produced through a dedicated dense layer and optimized using a task-specific loss function. This design allows the model to jointly learn both classification objectives within a single, efficient end-to-end architecture.

3.4. Multi-Task Loss-Based Training Procedure

The model training process follows the workflow illustrated in Fig. 3, which consists of three main stages: batch sampling, training step, and loss computation, followed by an evaluation of the model's capability on testing datasets.

During the batch sampling stage, a subset of flows is randomly selected from the training set. These flows are then organized into packet sequences using topology-constrained shuffling we explained before. Then, the resulting batches are processed in the training step, where each packet sequence is forwarded through the Stacked Dual-Path LSTM Encoder and the Inter-Path Aggregation Module. This process produces two prediction outputs: binary classification logits for anomaly detection and multi-class classification logits for attack type identification. Both outputs are subsequently evaluated using their respective loss functions.

The loss computation stage involves calculating and aggregate the Binary Prediction Loss and the Multi-Class Prediction Loss, both of which employ weighted Focal Loss to address class imbalance. The weighted Focal Loss is defined as:

$$\mathcal{L} = -\sum_{i=1}^{n} \alpha_i y_i (1 - \hat{y_i})^{\gamma} \log(\hat{y_i}) \tag{9}$$

In equation (9), α_i denotes the loss weight for class i which on our experiment setting defined as the inverse of square root number of samples belonging to that class. While $\hat{y_i}$ and y_i each corresponds to predicted class probability and actual class label from dataset, respectively.

4. Experiments & Result

Data preprocessing and feature selection were conducted in a Google Colab Pro environment without GPU acceleration. Deep learning model training was performed in a Kaggle environment using an NVIDIA Tesla P100 GPU. Further details regarding the datasets, experimental configuration, and evaluation results are provided in the following subsections.

4.1. Dataset

The primary dataset used in this study is Edge-IIoTset [35], [36], which consists of network traffic captures in PCAP format. The dataset contains 14 attack categories (excluding the normal class) and comprises over 21 million packets distributed across more than 11 million network flows. Furthermore, this dataset exhibits severe class imbalance: the dominant class (normal traffic flows) accounts for more than 50% of the samples, while minority classes such as MITM (Man-in-the-Middle) and Fingerprinting attacks each represent less than 0.01% of the data, and a there are 8 class that represented by less than 1% of the data. Feature selection was performed based on domain knowledge, and relevant packet fields were extracted using the Wireshark network analysis tool [37].

Packet-to-flow assignment was conducted by grouping packets according to the tcp.stream and udp.stream identifiers. For ARP traffic, which lacks transport-layer stream identifiers, packets were grouped based on source and destination IP addresses. For training and model evaluation, the dataset was split at the flow level, with 20% of flows reserved for the test set and the remaining flows used for training and validation.

4.2. Models & Training Configuration

The aim of this study is to conduct a comparative analysis of different model configurations. Two hyperparameters are considered as experimental variables: (1) the depth of the LSTM layers; and (2) the aggregation function used to combine the outputs of the binary and multi-class prediction paths (see Fig. 2). We also train a baseline model that consist of stacked LSTM Block with only multi-class classification objective. A summary of the model configurations, including the total number of parameters, is presented in Table 2. Our baseline model (with id: BASE) are configured so that the number of trainable parameters are comparable with every other models. Model identifiers are also provided to facilitate reference during the results analysis.

Table 2: Experiment models configuration and total number of trainable parameters.

Model ID	Aggregation	Layers Depth	Hidden Dimension	Number of Trainable Parameters
BASE	-	8	256	9 037 840
ADD-3	Addition	3	256	6 226 448
ADD-4	Addition	4	256	8 595 472
GAT-3	Gated	3	256	6 489 104
GAT-4	Gated	4	256	8 989 456
ATT-3	Attention	3	256	6 752 784
ATT-4	Attention	4	256	9 384 976

All models listed in Table 2 were trained using a consistent configuration to ensure a fair comparison. Each model was optimized with the AdamW optimizer [38] using a default weight decay of 0.01, which provides sufficient regularization as established in modern deep learning literature. We selected an initial learning rate of 2×10^{-6} for all experiments. The adaptive nature of AdamW makes the training procedure less sensitive to the initial learning rate choice compared to standard Stochastic Gradient Descent (SGD). Models were trained for 100 epochs with a batch size of 192, and the best-performing iteration was selected based on the lowest validation loss to ensure optimal convergence.

4.3. Evaluation Metrics

Two evaluation metrics were adopted to compare the experimental results. The first metric is the macro-averaged F1-score, defined as:

$$F_1 = \frac{2\ Pr\ Rc}{Pr + Rc} \tag{10}$$

The second metric is the macro-averaged area under the Precision-Recall curve (AUC-PR), which is formally defined as:

$$AUC\text{-}PR = \int Pr(Rc)\ \mathrm{d}Rc \tag{11}$$

In equation (10) and (11), Pr and Rc each denote a precision and recall, respectively. recision measures the accuracy of positive predictions and is defined as the ratio of correctly predicted positive samples to all predicted positive samples. Recall measures the completeness of positive predictions and is defined as the ratio of correctly predicted positive samples to all actual positive samples. Formally, computation of precision and recall are defined on equation (12):

$$\begin{aligned} Pr &= \frac{TP}{TP + FP} \\ Rc &= \frac{TP}{TP + FN} \end{aligned} \tag{12}$$

Where *TP*, *FP*, and *FN* represent the number of true positives, false positives, and false negatives, respectively. Macro-averaged F1-score and AUC-PR are selected due to their robustness in evaluating model performance under class-imbalanced conditions, which is a critical requirement for intrusion detection systems.

4.4. Experimental Results

After the training and validation procedures, the best-performing model from each experiment was evaluated on the test set. A summary of the evaluation results is presented in Table 3. Compared with the baseline model, all proposed architectures achieve better performance on the multi-class classification task and significantly outperform the baseline in binary classification. This performance improvement can be attributed to the fact that the baseline

Table 3: Experiment results evaluated on test set. Every evaluation measurement are expressed as percentage point (%). For multi-class evaluation, the scores are aggregated with macro-aggregation. Binary F1-score results for the baseline method are computed by binarizing the model outputs. Columns 3 and 7 report the p-value assessing the statistical significance of the proposed model against the additive aggregation model with the same number of layers. Column 6 reports the p-value comparing the proposed model against the baseline.

Model ID	**Binary Prediction**		**Multi-class Prediction**			
	F1-score ↑	***p*-value** vs. ADD	**F1-score ↑**	**AUC-PR ↑**	***p*-value** vs. BASE	***p*-value** vs. ADD
BASE	79.41	-	28.65	30.04	-	-
ADD-3	97.24	-	31.02	35.59	< 0.0001	-
ADD-4	97.27	-	30.56	32.79	0.0007	-
GAT-3	**97.47**	0.0741	30.80	33.93	< 0.0001	0.2868
GAT-4	97.32	0.1329	32.81	38.64	< 0.0001	0.0059
ATT-3	97.22	0.7254	**34.25**	37.08	< 0.0001	0.0008
ATT-4	97.15	0.6932	34.02	**40.08**	< 0.0001	0.0002

model is trained solely to minimize a multi-class loss, whereas the proposed multi-task architectures employ separate classification heads optimized specifically for each objective.

Overall, every models achieve comparably high binary F1-scores (above 97%, see Table 3), indicating that distinguishing normal versus malicious traffic is not a difficult task. In contrast, multi-class performance shows substantial variation. Attention-based models (ATT-3 and ATT-4) consistently outperform ADD and GAT variants in both macro F1-score and AUC-PR, achieving consistently higher macro F1-score and AUC-PR under severe class imbalance compared to ADD and GAT variants. Notably, model with attention aggregation with 3 layers achieves the highest macro F1-score, while same model with 4 layers yields the best AUC-PR, indicating a trade-off between balanced per-class accuracy and ranking quality. To rigorously validate this observation, McNemar's tests were conducted between paired model variants with identical encoder depth. For the multi-class prediction, the attention-based models outperform both additive and gated variants (p-value < 0.05, see Table 3), confirming the robustness of the observed improvements. While for the binary classification, the results indicate no statistically significant differences across model pairs (p-value > 0.05), suggesting that binary intrusion detection performance is largely saturated.

To evaluate the impact of the post-LSTM ReLU activation, we compared our best performing models with attention aggregation and 4 layers depth against similar setup with the ReLU layer removed. Experimental results showed that removing the activation led to a decrease in multi-class F1 Score from 34.02% to 29.28%, and decrease in area under PR-curve from 40.08% to 34.69%. This suggests that the ReLU layer is essential to enhances non-linear feature across stacked recurrent layers before the final aggregation stage.

As shown in Fig. 4, all models exhibit degraded performance under extreme class imbalance. For minority classes such as MITM and Fingerprinting, none of the models produce positive predictions, which is reflected by near-diagonal precision-recall curves. In contrast, higher precision-recall performance is observed for majority classes, including Normal traffic, DDoS TCP SYN Flood, DDoS UDP Flood, and DDoS ICMP Flood attacks.

5. Discussion

This study investigates how architectural choices in multi-task temporal models, specifically inter-path aggregation mechanisms and encoder depth affect intrusion detection models performance in IoT network traffic. From last section, we could observe that all evaluated models achieve consistently high performance on coarse-grained anomaly detection, but their ability to discriminate between attack types varies substantially. The findings demonstrate that while binary intrusion detection is largely saturated, architectural choices substantially impact fine-grained attack discrimination under severe class imbalance.

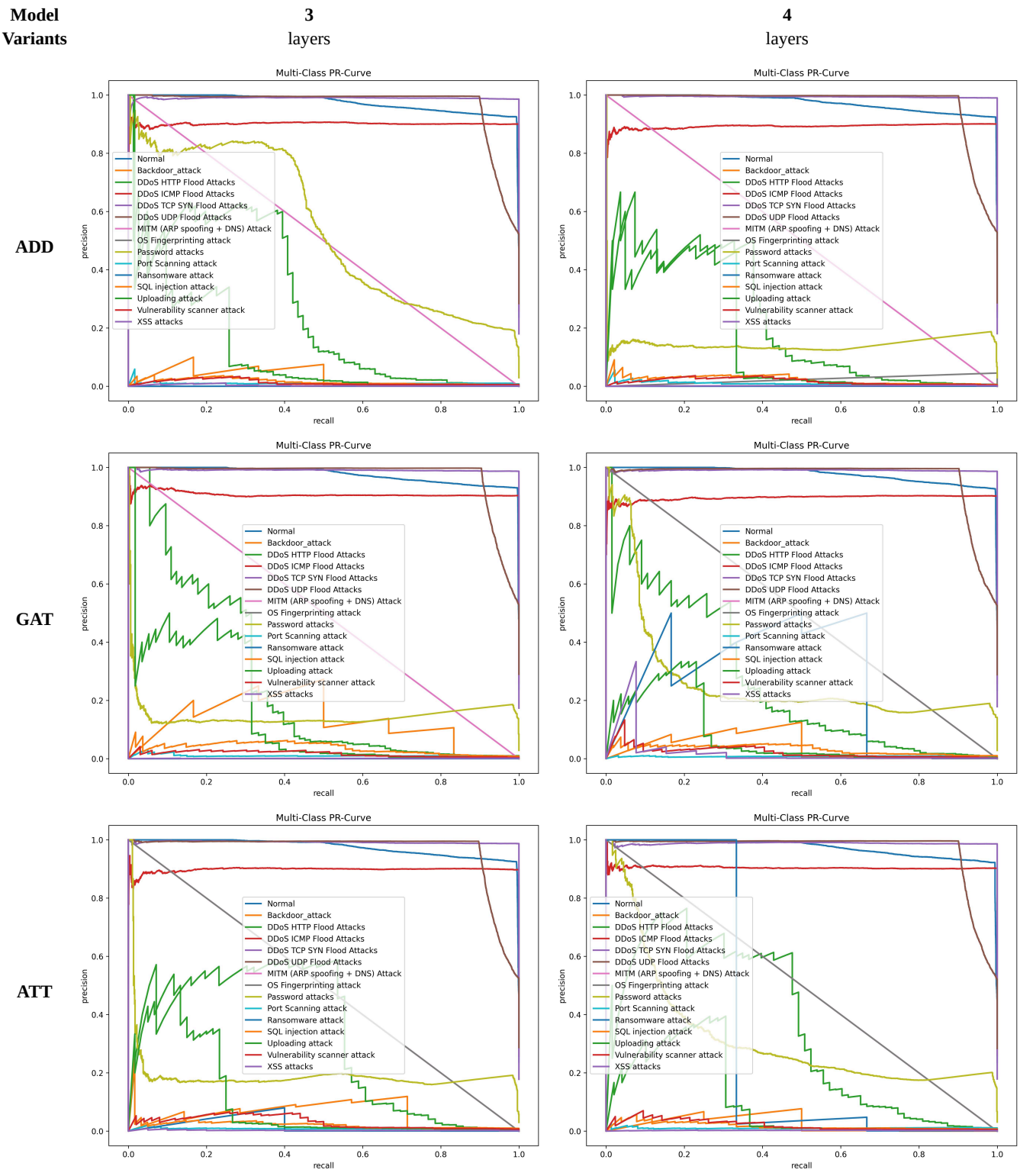

Fig. 4: Precision-Recall curves on the test set. The first column corresponds to models with three layers, whereas the second column corresponds to models with four layers. The top, middle, and bottom rows represent models with addition, gated, and attention aggregation functions, respectively.

From Table 3, it can be observed that attention-based aggregation consistently outperforms both additive and gated variants, even when using fewer layers and parameters (e.g., ATT-3 compared with ADD-4 and GAT-4 in Table 2). This indicates that selective, context-aware information transfer from the binary path is more effective than uniform or static fusion mechanisms for supporting multi-class attack classification. By dynamically weighting anomaly-relevant temporal features, the attention mechanism allows the multi-class path to focus on discriminative

subsequences that differentiate attacks with similar flow-level behavior. The comparison between three- and four-layer encoders reveals a depth-related trade-off between per-class balance and ranking quality. Shallower models tends to achieve higher macro F1-scores (supported by ADD and ATT variants), indicating more uniform performance across classes, while deeper encoders yield superior AUC-PR (supported by GAT and ATT variants), reflecting improved confidence calibration and ranking of positive samples. This suggests that deeper temporal abstraction benefits probability estimation but may amplify bias toward dominant classes, especially when minority samples are scarce.

Several limitations should be acknowledged. First, the extreme rarity of certain attack classes, such as MITM and Fingerprinting, prevents meaningful learning regardless of aggregation strategy, indicating that extreme class rarity imposes fundamental limits on purely loss-based re-weighting, and highlights the importance of complementary strategies such as data selection or rebalancing techniques, which remain outside the scope of this study. Second, flow construction relies on protocol-specific heuristics, which may limit generalization to alternative traffic representations. Third, the aggregation mechanism is designed as a unidirectional information transfer from binary classification to multi-class classification to avoid feature homogenization, where symmetric information exchange might cause the distinct temporal and structural features of each path to converge and become redundant, we acknowledge that this prevents the model from capturing potential bidirectional dependencies. Finally, evaluation is confined to a single dataset, restricting conclusions about cross-domain robustness. Nevertheless, the consistent gains from attention-based aggregation and topology-constrained shuffling provide strong evidence that structured temporal modeling and task-aware information sharing are critical for advancing IoT intrusion detection beyond binary classification.

6. Conclusion

In this study, we propose and evaluate several deep learning architectures for identifying attack behavior in IoT network traffic. The proposed models employ stacked LSTM layers to jointly address two objectives: binary classification (normal versus attack) and multi-class classification (attack type identification). We investigate the effects of varying encoder depth and inter-path aggregation strategies to facilitate knowledge sharing between the binary and multi-class prediction tasks. The experimental results indicate that context-aware information transfer from the binary path more effectively supports attack-type classification than static fusion mechanisms. In addition, to better approximate real-time network traffic conditions during training, a topology-constrained shuffling strategy is introduced, which improves model generalization to previously unseen temporal sequence patterns.

As stated in the previous section, this work is limited by the use of heuristic flow construction and evaluation on a single dataset, Edge-IIoTset. Accordingly, future work will incorporate richer structural information, such as explicit flow-level identifiers or learned flow embeddings, while extending the preprocessing module to handle a broader range of protocols beyond TCP and UDP. Furthermore, we intend to evaluate cross-dataset generalization to assess robustness under different traffic distributions, as well as conduct inference-time performance analysis to better characterize the suitability of the proposed models for real-time intrusion detection systems.

CRediT authorship contribution statement

C. B. Sabdana: Conceptualization, Methodology, Software, Formal analysis, Investigation, Data Curation, Writing – Review & Editing, Visualization. **N. D. Salyasari:** Conceptualization, Software, Investigation, Data Curation, Writing – Original Draft, Writing – Review & Editing, Visualization. **I. N. Z. Aliya:** Conceptualization, Software, Resources, Data Curation, Writing – Original Draft, Writing – Review & Editing, Visualization. **A. M. Shiddiqi:** Validation, Writing – Review & Editing, Supervision, Project Administration, Funding Acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The dataset was openly provided by IEEE DataPort [36] or could be downloaded from Kaggle here. The implementation of the proposed model and the experimental scripts used in this study are publicly available at: https://github.com/MaclaurinSeries/Multi-Task-Temporal-IDS.

Declaration of Generative AI and AI-assisted Technologies in The Writing Process

The authors used generative AI to improve the writing clarity of this paper. They reviewed and edited the AI-assisted content and take full responsibility for the final publication.

References

[1] F. Hussain, R. Hussain, S. A. Hassan, and E. Hossain, "Machine learning in IoT security: Current solutions and future challenges," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, pp. 1686–1721, 2020, doi: 10.1109/COMST.2020.2986444.

[2] M. S. Elksasy, "Understanding the internet of things (IoT) concepts, applications and standards: an overview," *Delta University Scientific Journal*, vol. 6, no. 1, pp. 205–210, 2023, doi: 10.21608/dusj.2023.291047.

[3] K. Kaur, A. Kaur, Y. Gulzar, and V. Gandhi, "Unveiling the core of IoT: comprehensive review on data security challenges and mitigation strategies," *Frontiers in Computer Science*, vol. 6, p. 1420680, 2024, doi: 10.3389/fcomp.2024.1420680.

[4] M. Țălu, "Security and privacy in the IIoT: threats, possible security countermeasures, and future challenges," *Computing&AI Connect*, vol. 2, no. 1, pp. 1–10, 2025.

[5] M. Țălu, "Securing IoT Ecosystems: Experimental Evaluation of Modern Lightweight Cryptographic Algorithms and Their Performance," *Journal of Cyber Security*, vol. 7, no. 1, pp. 565–587, 2025, doi: 10.32604/jcs.2025.073690.

[6] K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed, and M. Xu, "A survey on machine learning techniques for cyber security in the last decade," *IEEE access*, vol. 8, pp. 222310–222354, 2020, doi: 10.1109/ACCESS.2020.3041951.

[7] O. A. Wahab, "Intrusion detection in the iot under data and concept drifts: Online deep learning approach," *IEEE Internet of Things Journal*, vol. 9, no. 20, pp. 19706–19716, 2022, doi: 10.1109/JIOT.2022.3167005.

[8] M. A. Al-Garadi, A. Mohamed, A. K. Al-Ali, X. Du, I. Ali, and M. Guizani, "A survey of machine and deep learning methods for internet of things (IoT) security," *IEEE communications surveys & tutorials*, vol. 22, no. 3, pp. 1646–1685, 2020, doi: 10.1109/COMST.2020.2988293.

[9] S. Jamshidi, A. Nikanjam, N. K. Wazed, and F. Khomh, "Leveraging Machine Learning Techniques in Intrusion Detection Systems for Internet of Things," *arXiv preprint arXiv:2504.07220*, 2025.

[10] A. A. Ghani and S. A. Alasadi, "A Comprehensive Review: Intrusion Detection System Using Machine Learning in Internet of Things," in *2024 International Conference on Intelligent Cybernetics Technology & Applications (ICICyTA)*, 2024, pp. 290–295.

[11] M. Abdel-Basset, H. Hawash, R. K. Chakrabortty, and M. J. Ryan, "Semi-supervised spatiotemporal deep learning for intrusions detection in IoT networks," *IEEE Internet of Things Journal*, vol. 8, no. 15, pp. 12251–12265, 2021, doi: 10.1109/JIOT.2021.3060878.

[12] K. He, D. D. Kim, and M. R. Asghar, "Adversarial machine learning for network intrusion detection systems: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 25, no. 1, pp. 538–566, 2023, doi: 10.1109/COMST.2022.3233793.

[13] U. C. Akuthota and L. Bhargava, "The role of machine and deep learning in modern intrusion detection systems: A comprehensive review," *Computers and Electrical Engineering*, vol. 124, p. 110318, 2025, doi: 10.1016/j.compeleceng.2025.110318.

[14] S. Ismail, S. Dandan, and A. Qushou, "Intrusion Detection in IoT and IIoT: Comparing Lightweight Machine Learning Techniques Using TON_IoT, WUSTL-IIOT-2021, and EdgeIIoTset Datasets," *IEEE Access*, 2025, doi: 10.1109/ACCESS.2025.3554083.

[15] S. Baisthakur and B. Fitzgerald, "Multi-task learning long short-term memory model to emulate wind turbine blade dynamics," *Wind Energy Science Discussions*, vol. 2024, pp. 1–35, 2024.

[16] A. Aldhaheri, F. Alwahedi, M. A. Ferrag, and A. Battah, "Deep learning for cyber threat detection in IoT networks: A review," *Internet of Things and cyber-physical systems*, vol. 4, pp. 110–128, 2024, doi: 10.1016/j.iotcps.2023.09.003.

[17] Y. Zhang and Q. Yang, "A survey on multi-task learning," *IEEE transactions on knowledge and data engineering*, vol. 34, no. 12, pp. 5586–5609, 2021, doi: 10.1109/TKDE.2021.3070203.

[18] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *Ieee Access*, vol. 5, pp. 21954–21961, 2017, doi: 10.1109/ACCESS.2017.2762418.

[19] N. Shone, T. N. Ngoc, V. D. Phai, and Q. Shi, "A deep learning approach to network intrusion detection," *IEEE transactions on emerging topics in computational intelligence*, vol. 2, no. 1, pp. 41–50, 2018, doi: 10.1109/TETCI.2017.2772792.

[20] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE access*, vol. 7, pp. 41525–41550, 2019, doi: 10.1109/ACCESS.2019.2895334.

[21] S. A. Albelwi, "An intrusion detection system for identifying simultaneous attacks using multi-task learning and deep learning," in *2022 2nd International Conference on Computing and Information Technology (ICCIT)*, 2022, pp. 349–353. doi: 10.1109/ICCIT52419.2022.9711630.

[22] B. Sharma, L. Sharma, C. Lal, and S. Roy, "Anomaly based network intrusion detection for IoT attacks using deep learning technique," *Computers and Electrical Engineering*, vol. 107, p. 108626, 2023, doi: 10.1016/j.compeleceng.2023.108626.

[23] M. Abd Elaziz, M. A. Al-qaness, A. Dahou, R. A. Ibrahim, and A. A. Abd El-Latif, "Intrusion detection approach for cloud and IoT environments using deep learning and Capuchin Search Algorithm," *Advances in Engineering Software*, vol. 176, p. 103402, 2023, doi: 10.1016/j.advengsoft.2022.103402.

[24] R. Lazzarini, H. Tianfield, and V. Charissis, "A stacking ensemble of deep learning models for IoT intrusion detection," *Knowledge-Based Systems*, vol. 279, p. 110941, 2023, doi: 10.1016/j.knosys.2023.110941.

[25] P. Sanju, "Enhancing intrusion detection in IoT systems: A hybrid metaheuristics-deep learning approach with ensemble of recurrent neural networks," *Journal of Engineering Research*, vol. 11, no. 4, pp. 356–361, 2023, doi: 10.1016/j.jer.2023.100122.

[26] R. A. Elsayed, R. A. Hamada, M. I. Abdalla, and S. A. Elsaid, "Securing IoT and SDN systems using deep-learning based automatic intrusion detection," *Ain Shams Engineering Journal*, vol. 14, no. 10, p. 102211, 2023, doi: 10.1016/j.asej.2023.102211.

[27] H. Nandanwar and R. Katarya, “Deep learning enabled intrusion detection system for Industrial IOT environment,” *Expert Systems with Applications*, vol. 249, p. 123808, 2024, doi: 10.1016/j.eswa.2024.123808.
[28] Z. Z. Darban, G. I. Webb, S. Pan, C. C. Aggarwal, and M. Salehi, “CARLA: Self-supervised contrastive representation learning for time series anomaly detection,” *Pattern Recognition*, vol. 157, p. 110874, 2025, doi: 10.1016/j.patcog.2024.110874.
[29] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, “A detailed analysis of the KDD CUP 99 data set,” in *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, 2009, pp. 1–6. doi: 10.1109/CISDA.2009.5356528.
[30] N. Moustafa and J. Slay, “UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set),” in *2015 Military Communications and Information Systems Conference (MilCIS)*, 2015, pp. 1–6. doi: 10.1109/MilCIS.2015.7348942.
[31] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, “A survey of network-based intrusion detection data sets,” *Computers & Security*, vol. 86, pp. 147–167, 2019, doi: 10.1016/j.cose.2019.06.005.
[32] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, “Deep learning for cybersecurity intrusion detection: Approaches, datasets, and comparative study,” *Future Internet*, vol. 12, no. 4, p. 67, 2020, doi: 10.3390/fi12040067.
[33] J. L. Ba, J. R. Kiros, and G. E. Hinton, “Layer Normalization,” *arXiv preprint arXiv:1607.06450*, 2016, doi: 10.48550/arXiv.1607.06450.
[34] A. Vaswani *et al.*, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017, doi: 10.48550/arXiv.1706.03762.
[35] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, “Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning,” *IEEe Access*, vol. 10, pp. 40281–40306, 2022, doi: 10.1109/ACCESS.2022.3165809.
[36] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, “Edge-IIoTset: A New Comprehensive Realistic Cyber Security Dataset of IoT and IIoT Applications: Centralized and Federated Learning.” [Online]. Available: https://dx.doi.org/10.21227/mbc1-1h68
[37] [Online]. Available: https://wireshark.com/
[38] I. Loshchilov and F. Hutter, “Decoupled weight decay regularization,” *arXiv preprint arXiv:1711.05101*, 2017, doi: 10.48550/arXiv.1711.05101.