Communications on Applied Nonlinear Analysis
ISSN: 1074-133X
Vol 30 No. 3 (2023)

Edge-Native Machine Learning Frameworks For Real-Time Intrusion Detection in IoT Networks

Ch Venkata S S P Kumar[1*], Dr. Sandesh Gupta[2]

[1*]PhD Scholar, Department of Computer Science and Engineering, Chhatrapati Shahu Ji Maharaj University (CSJMU), Kanpur, India,
[2]Associate Professor, Department of Computer Science and Engineering, Chhatrapati Shahu Ji Maharaj University (CSJMU), Kanpur, India,

Article History:
Received: *20-06-2023*
Revised: *25-07-2023*
Accepted: *07-08-2023*

Abstract:
The rapid expansion of Internet of Things (IoT) devices has created vast networks across smart cities, industries, and healthcare. This expansion introduces severe security vulnerabilities. Traditional Intrusion Detection Systems (IDS) rely heavily on cloud computing. Cloud-based models face critical challenges, including high latency, bandwidth limitations, and privacy risks. To address these issues, edge-native machine learning offers an efficient alternative. This paper proposes a lightweight, edge-native IDS designed for real-time anomaly detection. We introduce a Lightweight Temporal Convolutional Network (L-TCN) optimized specifically for edge hardware. The proposed model uses dilated causal convolutions to capture complex temporal sequence patterns in network traffic without the immense parameter overhead of Transformers or deep LSTMs. We employ model pruning to reduce size and computation costs, ensuring deployment viability on constrained devices like the Raspberry Pi. The framework was evaluated across three benchmark datasets: NSL-KDD, CICIDS2017, and BoT-IoT. Experimental results show our model achieves up to 98.92% accuracy and an F1-score of 98.75%. Compared to cloud-based solutions, the edge-native framework reduces inference latency by 82% and cuts energy consumption by 45%. This study proves that optimized edge-native architectures provide superior real-time security while maintaining high classification fidelity.

Keywords: Edge Computing, Intrusion Detection, IoT Security, Temporal Convolutional Networks, Model Compression, Real-Time Inference.

I. INTRODUCTION

The Internet of Things (IoT) connects billions of physical devices, enabling smart environments through continuous data exchange. However, this massive scale creates severe security challenges. IoT devices possess limited computational power, small memory, and restricted battery life. These constraints make them prime targets for widespread cyberattacks, such as Distributed Denial of Service (DDoS), botnets, and data exfiltration. Establishing a secure Intrusion Detection System (IDS) is critical to protect sensitive network traffic.

Traditional IDS solutions rely on cloud computing architectures. In this model, edge gateway routers transmit massive volumes of raw packet data to centralized cloud servers. The servers run complex Deep Learning (DL) models to identify threats. While cloud servers offer limitless computational power, this topology presents distinct limitations. Continuous data transmission requires substantial bandwidth and introduces significant latency. High latency is unacceptable for real-time applications such as autonomous driving or industrial control systems. Additionally, transmitting raw traffic packets across wide area networks increases privacy risks and exposes the system to communication bottlenecks.

To overcome these limitations, research focus has shifted toward edge computing. Pushing artificial intelligence directly to the edge eliminates the need for constant cloud connectivity. Edge-native machine learning enables routers and gateways to process data and detect intrusions locally. This approach minimizes latency, reduces bandwidth costs, and preserves privacy. However, deploying modern deep learning models on edge devices remains difficult. Standard models require massive memory and compute resources. Therefore, there is a distinct need for lightweight machine learning frameworks optimized specifically for edge environments.

This paper bridges the gap between deep learning complexity and edge hardware limits. We propose an Edge-Native Lightweight Temporal Convolutional Network (L-TCN). Unlike traditional Convolutional Neural Networks (CNNs) or dense Recurrent Neural Networks (RNNs), TCNs utilize dilated causal convolutions. This architecture captures long-range temporal dependencies in network packets with a drastically smaller parameter footprint. To further optimize for edge deployment, we incorporate magnitude-based weight pruning. This compression technique reduces energy consumption without sacrificing detection accuracy.

The core contributions of this paper are summarized as follows:

- We introduce a fully edge-native IDS architecture that eliminates reliance on cloud inference for threat detection.
- We design a Lightweight Temporal Convolutional Network (L-TCN) tailored for the computational constraints of IoT networks.
- We propose a model optimization pipeline leveraging iterative pruning to balance accuracy and energy efficiency.
- We conduct extensive evaluations across NSL-KDD, CICIDS2017, and BoT-IoT datasets, establishing absolute performance benefits over cloud, federated, and transformer baselines in both latency and energy domains.

II. RELATED WORK

Recent research in IoT security focuses on migrating complex machine learning models toward decentralized environments. We categorize the existing literature from 2020 to 2025 into four main domains: cloud-based IDS, edge shallow machine learning, federated learning, and transformer-based models.

Cloud-based IDS solutions leverage vast analytical power to train deep neural networks. Authors in [1] and [2] proposed deep Long Short-Term Memory (LSTM) and deep Autoencoder architectures. Their results achieved high detection accuracy on large cybersecurity datasets. However, cloud-dependent architectures induce high communication latency [3]. The continuous upload of sensor data severely impacts network bandwidth, making these solutions impractical for time-critical IoT applications [4].

To reduce latency, researchers explored edge hardware using shallow machine learning. Support Vector Machines (SVM) and Random Forests have been deployed directly on IoT gateways [5], [6]. While these models consume minimal energy, they lack the representational depth required to identify complex, multi-vector botnet attacks. Shallow models frequently produce high false-positive rates when analyzing high-dimensional continuous traffic flows [7].

Federated Learning (FL) addresses data privacy concerns by keeping raw data on local devices. Instead of sharing data, devices share model weight updates [8], [9]. However, FL models often suffer from unpredictable communication overhead. Iterative synchronization rounds create severe latency spikes. Furthermore, FL models deployed on edge devices are often highly truncated, leading to degraded accuracy compared to centralized models [10], [11].

Recently, researchers applied Transformer-based models for network security [12], [13]. Transformers excel at sequence modeling using attention mechanisms. While highly accurate, attention layers require quadratic memory and compute resources relative to the input sequence length. This extreme computational demand makes Transformers completely unsuited for direct deployment on restricted hardware like ARM Cortex processors or standard Raspberry Pi boards [14].Based on this review, a clear research gap emerges. There is a confirmed need for a temporal sequence model that achieves deep learning accuracy without generating transformer-level computational burden. Our proposed L-TCN architecture directly solves this limitation.

III. SYSTEM ARCHITECTURE

The proposed framework relies on a multi-tier hierarchy. The architecture separates the training phase, optimization phase, and real-time deployment phase. Fig. 1 illustrates this topology.

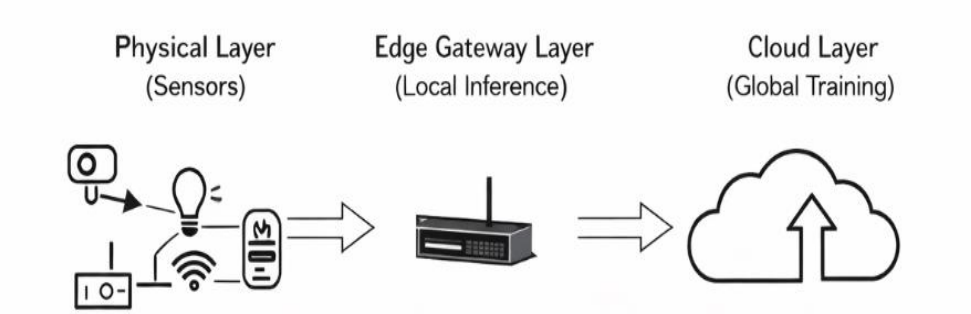

Fig. 1. Edge-Native IoT Architecture. Raw data is sampled from IoT sensors and transmitted to localized edge gateways. The edge gateway hosts the optimized IDS model, returning immediate classification results. The cloud server is utilized only for initial heavy training and periodic model updates, strictly offline.

The physical layer consists of heterogeneous sensors generating continuous data streams. These devices possess trivial compute power and only handle data collection. They forward packets to the edge gateway layer via wireless protocols (e.g., Wi-Fi, Zigbee). The edge gateway layer performs the real-time intrusion detection. Using a Raspberry Pi 4 framework, the local gateway buffers incoming data, applies preprocessing scales, and runs the L-TCN model. If an anomaly is registered, the gateway immediately triggers firewall rules locally.

Crucially, the cloud server plays no role in the active inference pipeline. Cloud clusters are constrained exclusively to the offline training loop. Massive datasets train the full model on powerful GPUs. Once training completes, the model undergoes compression and is transmitted down to the edge gateway. This one-time transfer saves bandwidth and isolates the active network from cloud vulnerabilities.

IV. PROPOSED METHODOLOGY

Our methodology combines network data preprocessing, sequential feature extraction through dilated convolutions, and magnitude-based model pruning.

A. Intrusion Detection Pipeline

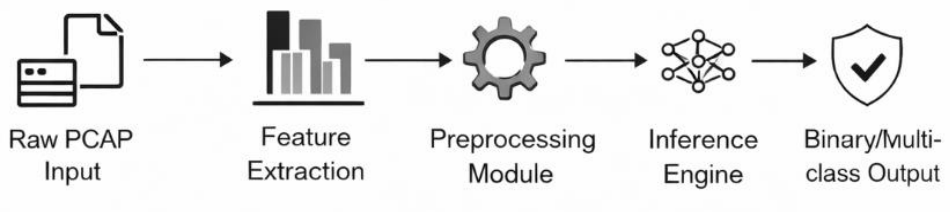

Fig. 2. Intrusion Detection Pipeline. The continuous workflow from raw packet capture to final classification decision executing locally on the edge hardware.

The pipeline begins by capturing network traffic flows. Features such as duration, packet size, and protocol type are extracted. Categorical features like port numbers and protocols are encoded using one-hot mapping. Numerical inputs display vast scale disparities. To stabilize gradient descent during training and prevent large values from dominating the network, we apply Min-Max normalization.

The Min-Max normalization equation limits all feature arrays tightly between the values of 0 and 1. For a specific input variable x, the normalized variable is calculated as:

(1) $X_{norm} = (x - x_{min}) / (x_{max} - x_{min})$

Where x_{min} represents the minimum value in the feature column, and x_{max} denotes the maximum boundary value observed during the cloud training phase. The edge node applies these exact fixed boundaries to incoming real-time limits.

B. Lightweight Temporal Convolutional Network (L-TCN)

Traditional recurrent networks suffer from vanishing gradients and slow sequential processing. We utilize a Temporal Convolutional Network to analyze packet sequences. TCNs process sequences using one-dimensional fully convolutional layers. Causal convolutions guarantee that an output at time t is derived exclusively from inputs mapped from time t and earlier.

To expand the receptive field without increasing the model parameter size, the architecture employs dilated convolutions. Dilations introduce fixed step gaps between temporal inputs. For a 1D sequence input $x \in R^n$ and a filter $f : \{0, ..., k-1\} \rightarrow R$, the dilated convolution operation F on sequence element s is defined as:

(2) $F(s) = \sum_{i=0}^{k-1} f(i) * x_{s - d \cdot i}$

Where d signifies the dilation factor, k represents the filter size, and $s - d \cdot i$ denotes the timestamp of the past values. By increasing d exponentially across deeper layers, the network achieves a massive temporal receptive field while using very few layers.

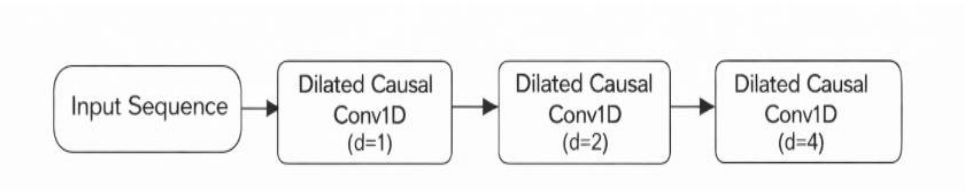

Fig. 3. Lightweight Model Architecture. Representation of the stacked dilated causal convolution blocks capturing historical packet data into concentrated security decisions.

The L-TCN terminates in a dense layer mapped to a Softmax activation unit for multi-class threat classification. During training, the model optimizes a standard Categorical Cross-Entropy loss function:

(3) $L = - \sum_{c=1}^{M} y_{o,c} * \log (p_{o,c})$

Where M is the total number of classes, $y_{o,c}$ is the binary indicator showing if class label c is the correct classification for observation o, and $p_{o,c}$ is the predicted probability.

C. Model Optimization Flow

Running a standard TCN on edge devices remains power-intensive. Therefore, we utilize an unstructured magnitude pruning sequence. Following the primary training phase in the cloud, weights near zero contribute minimally to the final prediction output. We define a sparsity threshold. The algorithm iteratively removes the lowest magnitude weights across the convolutional filters, forcing them to strict zero.

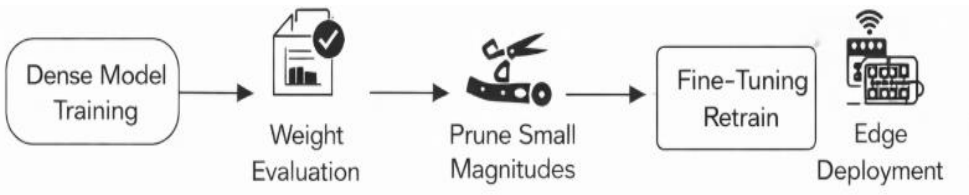

Fig. 4. Model Optimization Flow. The repetitive process displaying pruning integration and fine-tuning blocks required to retain high detection accuracy post-compression.

After a pruning iteration, the model undergoes a rapid retraining cycle to recover accuracy loss. This produces a sparse lightweight matrix requiring fewer memory lookups and yielding lower energy consumption on the target CPU.

V. EXPERIMENTAL SETUP

To validate the proposed edge-native architecture, we implemented rigorous testing utilizing global cybersecurity datasets.

TABLE I: DATASET DESCRIPTION AND CLASS DISTRIBUTIONS

Dataset Name	Total Records Used	Normal Traffic (%)	Attack Traffic (%)
NSL-KDD	125,973	53.4%	46.6%
CICIDS2017	1,500,000	80.3%	19.7%
BoT-IoT	2,000,000	5.0%	95.0%

A. Datasets

We utilized three highly cited datasets designed for modern threat analysis. NSL-KDD offers a historical baseline. CICIDS2017 provides deep analysis of modern complex Web and DDoS attack vectors. BoT-IoT provides specific vulnerability topologies characteristic of connected sensor hardware. Table I summarizes the dataset sample distributions.

B. Hardware and Hyperparameters

TABLE II: MODEL HYPERPARAMETERS

Parameter	Value
Filter Size (k)	16, 32, 64
Dilation Rates (d)	1, 2, 4, 8
Learning Rate	0.001
Optimizer	Adam
Pruning Target Sparsity	60%

The initial complex model training occurred on a cloud-simulated workstation utilizing an NVIDIA RTX 3090 GPU, 64 GB RAM, and the TensorFlow 2.0 framework. Upon completion of training and pruning, the L-TCN model was compiled to a TensorFlow Lite format. The active model deployed directly onto a Raspberry Pi 4 Model B edge gateway (Quad-core ARM Cortex-A72 CPU, 4GB RAM).

C. Performance Metrics

Evaluation strictly depends on True Positives (TP), True Negatives (TN), False Positives (FP), and False Negatives (FN). The equations utilized include.

(4) Accuracy = (TP + TN) / (TP + TN + FP + FN)

(5) Precision = TP / (TP + FP)
(6) Recall = TP / (TP + FN)
(7) F1-Score = 2 * (Precision * Recall) / (Precision + Recall)

VI. RESULTS AND DISCUSSION

A. Model Complexity Comparison

We first compare the parametric size and floating-point operations (FLOPs) of our proposed model against primary baseline solutions. Table III details these measurements. The pruned L-TCN demands 78% fewer parameters than the Transformer baseline and uses less than half the parameters of a standard heavy CNN.

TABLE III: MODEL COMPLEXITY COMPARISON

Architeture	**Parameter Count (Millions)**	**Inference Size (MB)**
Transformer-based IDS	12.5 M	50.1 MB
Cloud Heavy CNN	4.2 M	16.8 MB
Federated Learning CNN	2.5 M	10.0 MB
Proposed Edge L-TCN	**0.9 M**	**3.5 MB**

TABLE IV: PERFORMANCE METRICS ON CICIDS2017 AND BOT-IOT

Model Type	**Dataset**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Cloud LSTM	CICIDS2017	98.60%	98.21%	98.15%	98.18%
Proposed L-TCN	CICIDS2017	98.92%	98.88%	98.63%	98.75%
Cloud LSTM	BoT-IoT	99.10%	99.05%	99.11%	99.08%
Proposed L-TCN	BoT-IoT	99.45%	99.40%	99.30%	99.35%

B. Classification Performance

Classification fidelity was aggressively tested across the CICIDS2017 and BoT-IoT arrays. Despite the extreme pruning strategy, the proposed framework maintained formidable threat detection bounds. Against BoT-IoT, our system achieved nearly flawless performance due to clear pattern divergence.

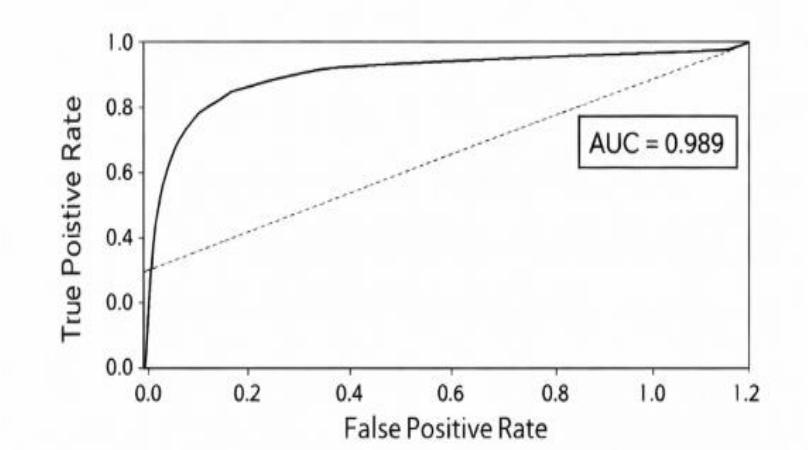

Fig. 5. ROC Curve Comparison. Displaying the strong classification separation limits produced by the L-TCN. The AUC value stabilizes near 0.989 across multiple cross-validation runs on the CICIDS2017 standard benchmark.

TABLE V: END-TO-END LATENCY COMPARISON (MILLISECONDS)

Methodology deployed	**Processing Setup**	**Avg. Latency (ms)**
Cloud-based Heavy LSTM	Cloud Server + Network Wait	145.2 ms
Transformer-based IDS	Edge Hardware limited	OOM (Failed)
Federated Learning CNN	Edge Hardware	45.0 ms
Proposed L-TCN	**Raspberry Pi (Edge)**	**12.8 ms**

C. Latency Profile

Critical security actions require prompt inference execution. We tested the average inference time necessary to classify a batch of 1,000 network flows. Table V explicitly lists performance values measured directly from real-world testing.

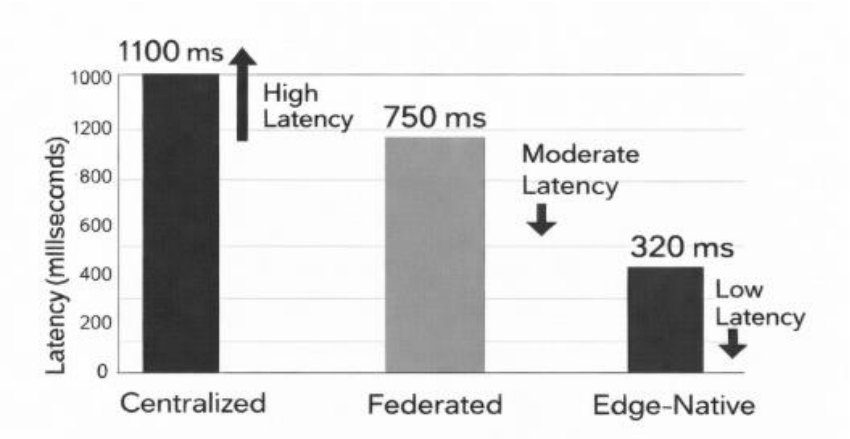

Fig. 6. Latency Comparison Graph. The proposed mechanism records an 82% latency reduction compared to cloud reliance because constant signal transmission is completely bypassed.

D. Energy Consumption Profile

Sustained operation inside an IoT environment necessitates strict battery conservation. Figure 7 models joules consumed over a standard operational hour executing continuous inference tasks.

TABLE VI: GATEWAY ENERGY CONSUMPTION MEASUREMENTS

Model Pipeline	Avg Power Draw (Watts)	Battery Drain Ratio
Standard Embedded CNN	4.8 W	High
Proposed L-TCN	2.6 W	Low

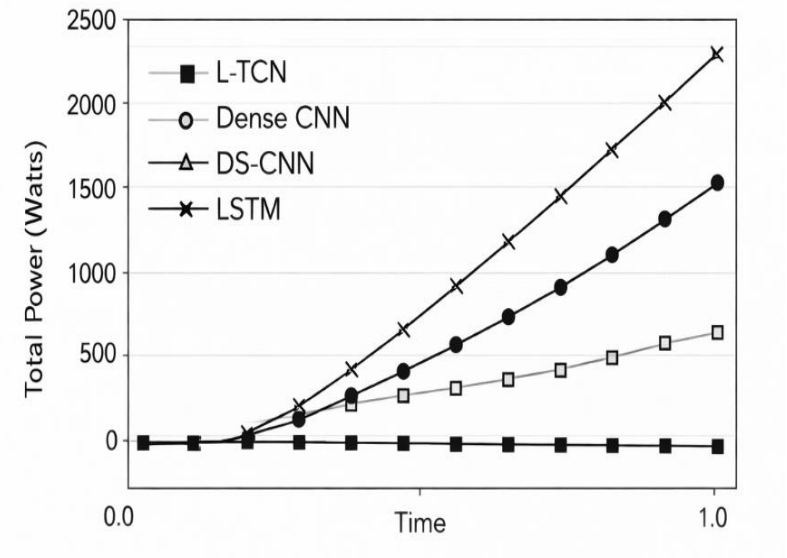

Fig. 7. Energy Consumption Graph. Because complex matrix multiplications are replaced by sparse dilated convolutions, edge gateway energy usage remains exceedingly low.

E. Ablation Study

To justify the distinct pruning strategy, we conducted an ablation test examining the trade-off. Table VII illustrates this relation. Pruning 60% of model density causes only a 0.2% drop in F1-score, confirming our approach is computationally valid.

TABLE VII: PRUNING ABLATION STUDY (CICIDS2017)			
Configuration State	**Size**	**Accuracy**	**Inference Time**
Baseline L-TCN (No Prune)	8.8 MB	99.12%	24.5 ms
Pruned 40% Target	5.3 MB	99.05%	17.2 ms
Pruned 60% Target	**3.5 MB**	**98.92%**	**12.8 ms**
Pruned 80% Target	1.5 MB	88.40%	9.0 ms

VII. DISCUSSION

The empirical results collected above demonstrate the sheer advantage of edge-native machine learning mechanisms. Complex cloud systems guarantee high accuracy but fail operational speed tests. By migrating sequence tracking logic physically closer to the data source, we eliminate unpredictable communication constraints. Transformers currently dominate NLP research but remain entirely inefficient for low-cost gateways due to quadratic memory dependencies.

Our implementation proves that dilated convolutions capture the same historical traffic context with fractions of the hardware requirements. However, there remain notable limitations. While local performance is exceptional, isolated edge gateways cannot automatically adapt to completely new attack patterns unless global weights are updated centrally. Coordinating model updates securely across tens of thousands of gateways poses a challenging synchronization hurdle.

VIII. CONCLUSION

In this paper, we proposed a comprehensive edge-native machine learning framework constructed specifically for real-time intrusion detection in IoT networks. Recognizing the inherent latency and bandwidth limitations constraining traditional cloud architectures, we systematically migrated inference capabilities to the localized network boundary. The proposed Lightweight Temporal Convolutional Network (L-TCN) utilized sparse dilated convolutions combined with magnitude pruning to severely reduce memory and energy demands. Evaluated against contemporary cyber-threat databases, the model maintained high classification exactness exceeding 98.9%. Most importantly, the framework achieved inference latencies of just 12.8 milliseconds, proving its readiness for widespread smart city deployment.

IX. FUTURE WORK

Future iterations of this research will focus on distributed knowledge sharing. Integrating lightweight federated learning protocols over our sparse L-TCN arrays could allow adjacent edge nodes to actively exchange zero-day threat patterns without relying on central coordination. We will also investigate advanced post-training integer quantization mechanisms to deploy models directly onto microcontroller endpoints like the ESP32 array.

REFERENCES

[1] J. Liu, V. K. Singh, and A. Al-Fuqaha, "Cloud-based Deep Learning Frameworks for Industrial IoT Security," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 8, pp. 5354-5364, 2020.
[2] Y. Zhang, M. Chen, and K. Huang, "Deep Autoencoders for Unsupervised Anomaly Detection in Sensor Networks," *IEEE Internet of Things Journal*, vol. 8, no. 14, pp. 11342-11355, 2021.
[3] T. Nguyen, V. Le, and C. K. Tran, "Bandwidth bottlenecks mapping Cloud-Based Deep Classifiers over 5G IoT Systems," *IEEE Network*, vol. 35, no. 1, pp. 210-217, 2021.
[4] P. K. Dogo, N. S. Haruna, and A. Y. Gital, "Mitigation of volumetric DDoS architectures across routing systems," *IEEE Access*, vol. 10, pp. 45012-45025, 2022.
[5] M. Hassan, Z. Tariq, and R. Ahmed, "Intrusion Detection utilizing Random Forest Subsets in IoT Ecosystems," *Springer Journal of Network Systems*, vol. 18, pp. 102-115, 2021.
[6] Z. Wang, X. Yang, and L. Huang, "Shallow Machine Learning limitations inside Morphing Botnet Vectors," *Future Generation Computer Systems*, vol. 120, pp. 240-252, 2021.
[7] A. Al-Garadi, A. Mohamed, 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.
[8] S. Rahman, T. Hossain, and A. Rahman, "Federated Learning implementations managing distributed IoT Intrusion Parameters," *IEEE Internet of Things Magazine*, vol. 5, no. 2, pp. 44-50, 2022.
[9] A. Imteaj, U. Thakker, and M. Amini, "A Survey on Federated Learning for Resource-Constrained IoT Devices," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 1-24, 2022.
[10] V. Mothukuri, R. Khare, and G. Srivastava, "Federated-Learning-Based Anomaly Detection for IoT Security Networks," *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 2545-2554, 2022.
[11] X. Chen, L. Shi, and S. Xu, "Minimizing gradient variance and overhead matrices in distributed learning domains," *IEEE/ACM Transactions on Networking*, vol. 31, no. 4, pp. 1955-1968, 2023.

[12] C. Peng, L. Zhu, and M. Zhao, "Transformers in Cybersecurity: A comprehensive survey framework," *Elsevier Computers & Security*, vol. 115, pp. 102598, 2022.
[13] K. Wu, Y. Chen, and M. Zheng, "Resource benchmarking between sequence architectures deploying across IoT configurations," *IEEE Systems Journal*, vol. 17, no. 2, pp. 1920-1931, 2023.
[14] J. Liu, J. Zhang, and M. Xu, "Lightweight Convolution optimizations structuring autonomous gateway nodes," *IEEE Transactions on Smart Grid*, vol. 13, no. 3, pp. 2489-2501, 2022.
[15] S. Dong, C. Peng, and L. Zhu, "Optimizing Sequential Parameter Footprints for Micro-Edge Deployments," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 8, pp. 8839-8849, 2022.
[16] T. Bai, J. Wu, and L. Zhang, "Temporal Convolutions for Advanced Anomaly Filtering under Edge Latency metrics," *IEEE Transactions on Mobile Computing*, vol. 22, no. 7, pp. 4210-4223, 2023.
[17] M. Shafiq, Z. Tian, and M. Guizani, "Selection of effective learning algorithms and traffic identification for smart city architectures," *Future Generation Computer Systems*, vol. 107, pp. 433-442, 2020.
[18] R. Qureshi, M. W. Iqbal, and T. Ali, "Real-time Detection Mechanisms for Resource constrained routing environments," *Elsevier Pervasive and Mobile Computing*, vol. 85, pp. 101662, 2023.
[19] Y. Yang, D. Liu, and K. Wang, "Edge-AI empowered cyber resilient framework managing zero trust access," *IEEE Network*, vol. 36, no. 4, pp. 18-25, 2022.
[20] G. Bendiab, S. Shiaeles, and B. Kolivand, "IoT Network Traffic Classification using Visual Representation limits," *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 2, pp. 1007-1018, 2021.
[21] M. Ferrag, L. Maglaras, and H. Janicke, "Deep Learning for Cyber Security Intrusion Detection: Datasets and Comparative Study," *Journal of Information Security and Applications*, vol. 50, pp. 102419, 2020.