Engineering Systems and Intelligent Technologies
ESIT
ISSN: 3071-253X/© 2026, The Authors. (CC BY 4.0) License.
Journal Homepage
https://pub.scientificirg.com/index.php/ESIT

Lightweight Federated Edge AI Framework for Real-Time Intrusion Detection in Industrial IoT Ecosystems

Ahmed Hammad[a,1], **and Hegazi M. Ibrahim**[b]

[a] *Department of Information Security and Cyberspace, Faculty of Information Technology, Isra University, Amman, Jordan;*
E-mail: ahmed.hammad@iu.edu.jo

[b] *Department of Computer Science, Faculty of Information Technology, Isra University, Amman, Jordan;*
E-mail: hegazi.ibrahim@iu.edu.jo

ABSTRACT

Industrial control networks connect programmable logic controllers (PLCs) to enterprise systems, which exposes closed-loop processes to flooding, scanning, spoofing, and injection attacks. Cloud-hosted intrusion detection can miss millisecond response requirements and requires the export of proprietary process data, whereas isolated edge detectors observe only local traffic. This paper presents Fed-EdgeIDS, a federated intrusion detection framework for low-power industrial gateways. Its classifier, DSA-Net, combines one-dimensional depthwise separable convolutions with four-head self-attention and has 67,831 trainable parameters, 94.6% fewer than a one-dimensional CNN baseline with 1,248,500 parameters. Each gateway clips its model update, adds Gaussian noise, and transmits the largest 30% of coordinates after unbiased 8-bit stochastic quantization with local error feedback. With four of 20 gateways sampled per round, one upload occupies 61,051 bytes, and 50 rounds require 11.6 MB of uplink payload against 952.5 MB for uncompressed federated averaging of the baseline. The server aggregates updates with a coordinate-wise trimmed mean and Nesterov momentum. Over five random seeds, Fed-EdgeIDS reaches 98.74% ± 0.12% accuracy on Edge-IIoTset and 99.02% ± 0.09% on CIC-IDS2017, with false alarm rates of 0.62% and 0.49%. The client-level privacy bound of the update perturbation is weak ($\varepsilon \approx 24.8$ at $\delta = 10^{-5}$), so the noise acts as obfuscation and not as a formal guarantee. On a Raspberry Pi 4 and an NVIDIA Jetson Nano, mean per-window inference takes 3.74 ms and 1.18 ms, and mean detection-to-containment-dispatch latency is 4.56 ms and 1.82 ms, excluding window accumulation and enforcement time.

PAPER INFORMATION

HISTORY

Received: 06 August 2026
Revised: 30 September 2026
Accepted: 15 October 2026
Online: 28 October 2026

MSC

68T07; 68M25; 68M14; 68P27

KEYWORDS

Industrial Internet of Things;
Edge Artificial Intelligence;
Federated Learning;
Intrusion Detection;
Model Quantization.

1. INTRODUCTION

Operational control of oil refineries, water treatment plants, and high-voltage substations depends on timely and predictable packet delivery. Field transmitters digitize pressure, temperature, and flow measurements and forward the readings to programmable logic controllers (PLCs) over industrial networks such as Modbus TCP, EtherCAT, and PROFINET [1], while MQTT serves as a lightweight messaging protocol for connected devices [2]. PLCs execute scan cycles of roughly 5

[1]Corresponding author: Department of Information Security and Cyberspace, Faculty of Information Technology, Isra University, Amman, Jordan; E-mail: ahmed.hammad@iu.edu.jo

to 20 ms to update proportional-integral-derivative (PID) routines that drive valves and motors. Delayed or lost frames degrade closed-loop stability and can force protective interlocks to act.

The Purdue reference model historically kept field equipment at Levels 0 and 1 separate from corporate systems at Level 4 through air gaps and point-to-point serial links. Industry 4.0 integration has weakened this separation, because field switches now exchange data with manufacturing execution systems, SCADA historians, and enterprise resource planning platforms. This integration exposes long-lived PLCs to routed IP traffic. Many of these controllers lack hardware cryptographic modules, firmware signature verification, and transport-layer authentication, so a syntactically valid command may be accepted without further checks.

Attacks on industrial control systems illustrate the consequences. Stuxnet altered centrifuge drive frequencies while replaying normal telemetry to operator displays [3]. Industroyer issued IEC 60870-5-104 and IEC 61850 commands to open circuit breakers at a Ukrainian substation [4], following the 2015 intrusions analyzed in [5]. TRITON interacted directly with a Safety Instrumented System at a petrochemical facility and caused a safety system shutdown [6]. At the data level, false data injection (FDI) can introduce arbitrary errors into state estimates of power grids without triggering existing bad-data detection [7].

Machine learning based intrusion detection systems (IDSs) have traditionally been deployed centrally, with switch mirror ports forwarding raw frames to a data center that hosts the classifier [8]. Four limitations hinder this design in production plants. First, protection functions require transfer times of a few milliseconds [9], whereas wide-area paths add tens to hundreds of milliseconds, so an actuator may already have moved when a remote server flags a command. Second, mirroring all frames of a large plant loads backhaul links. Third, raw process signals reveal batch schedules, reactor temperatures, and yields that operators are reluctant to export. Fourth, a severed uplink disables the detector.

Edge computing shifts computation toward the data source and addresses response time, bandwidth, and privacy concerns [10]. It is also regarded as a supporting technology of Industry 5.0 [11]. Distributed attack detection on fog nodes has been reported to outperform a centralized detector in a deep learning setting [12], and edge intelligence has become a broader research direction [13, 14]. A standalone gateway, however, observes only the subnet it monitors. A gateway on an intake pump never sees traffic from a packaging line or a boiler loop, so the resulting model overfits to local polling behavior and misses attack patterns first seen elsewhere.

Federated learning (FL) addresses this gap by letting gateways train a shared network on local data and exchange only model updates with a coordinating server [15]. Four obstacles arise on industrial edge networks. Local updates computed on skewed traffic diverge from the centralized solution and degrade federated averaging [16]. Gateway memory is limited, which restricts model size and motivates compression [17]. Uplinks are slower than downlinks and often bandwidth constrained [18]. Finally, shared gradients can leak private training data [19], and compromised participants can behave as Byzantine workers [20].

This paper addresses these obstacles within one edge-ready pipeline, named Fed-EdgeIDS. The main contributions are as follows.

1. **Compact classifier:** DSA-Net combines one-dimensional depthwise separable convolutions with a four-head scaled self-attention block. It has 67,831 trainable parameters, a 94.6% reduction relative to a one-dimensional CNN baseline with 1,248,500 parameters. The attention-free depthwise separable backbone has 38,519 parameters (96.9% reduction).

2. **Communication reduction:** Top-k sparsification (30%), unbiased 8-bit stochastic quantization, and a local residual buffer reduce one upload to 61,051 bytes. Over 50 rounds with four sampled gateways per round, the cumulative uplink payload is 11.6 MB, against 952.5 MB for uncompressed federated averaging of the baseline, a reduction of 98.8%. The compact model alone reduces the dense upload by 94.6%, and the sparse 8-bit packet is a further 77.5% smaller than the dense 32-bit DSA-Net upload.

3. **Update perturbation and robust aggregation:** Each gateway clips the L_2 norm of its update and adds Gaussian noise before upload, with Rényi accounting that bounds the resulting privacy loss at a weak client-level value. The server replaces plain averaging with a coordinate-wise trimmed mean and Nesterov momentum. The trimmed mean is inactive at the four updates aggregated per round, and its robustness effect is not evaluated in this study.

4. **Evaluation on public data and edge hardware:** The framework is evaluated over five random seeds on Edge-IIoTset (15 classes) and CIC-IDS2017 (8 classes) against five baselines, and is profiled on a Raspberry Pi 4 Model B and an NVIDIA Jetson Nano. Mean per-window inference takes 3.74 ± 0.08 ms and 1.18 ± 0.03 ms, mean detection-to-dispatch latency is 4.56 ± 0.10 ms and 1.82 ± 0.05 ms, and the Raspberry Pi deployment uses 148 MB of RAM.

The remainder of the paper is organized as follows. **Section 2** reviews industrial intrusion detection, federated learning for IoT security, and model compression. **Section 3** presents the architecture, threat model, DSA-Net, update perturbation,

quantization with error feedback, and trimmed-mean aggregation. **Section 4** describes the datasets, testbed, baselines, and classification results. **Section 5** reports ablations, hardware profiling, the noise-utility trade-off, and limitations. **Section 6** concludes the paper.

2. LITERATURE REVIEW

Rule-based engines such as Snort [21] and Bro [22] were the first line of defense in monitored networks. Bro passively monitors a network link and separates mechanism from policy through an event engine and a policy script interpreter [22]. Signature matching nevertheless has limits against zero-day attacks and against syntactically valid commands used maliciously. In a Modbus TCP network, an attacker does not need to malform any header. A well-formed write to a coil or register (function codes 0x05 and 0x06) passes signature rules unflagged, yet can drive the process into an unsafe state.

Statistical and shallow learning detectors flag deviations from established process behavior. Tai et al. studied random forests, gradient boosting, and neural networks for anomaly detection in industrial control systems [23]. Ahmad et al. compared support vector machines, random forests, and extreme learning machines for intrusion detection [24]. A study published in this journal compared five ensemble learners for man-in-the-middle detection on the TON_IoT dataset [25].

Deep networks learn hierarchical representations directly from flow records and headers [26]. Deep belief networks were applied to intrusion detection in [27], and one-dimensional CNNs extract local patterns from ordered signals [28]. Recurrent models capture multi-packet dependencies, including recurrent neural networks [29] and long short-term memory (LSTM) networks [30]. Deep models for malicious activity identification in industrial IoT were studied in [31], and a comparative study of deep learning based intrusion detection across datasets appears in [8]. Machine and deep learning methods for IoT security are surveyed in [32]. Such systems are normally trained centrally, which requires pooling raw traffic. Self-attention [33] has since been applied to intrusion detection, for example in the Transformer-based RTIDS [34], which is evaluated in a centralized setting.

Federated learning allows several sites to train a shared model without pooling raw data [35]. Rey et al. applied federated multilayer perceptron and autoencoder models to malware detection on the N-BaIoT dataset and compared several aggregation functions as countermeasures against poisoning participants [36], and Mothukuri et al. trained federated gated recurrent units for IoT attack detection [37]. Nishio and Yonetani proposed client selection to accommodate heterogeneous client resources [38], and Wang et al. tuned the number of local updates between global aggregations under a resource budget in edge computing systems [39]. These studies address learning accuracy, participation, resource allocation, or poisoning individually, whereas Fed-EdgeIDS combines a compact model, a sparse quantized uplink, update perturbation, and robust aggregation with measurements on physical gateways.

Running neural classifiers on small boards requires compression of both the model and the communication. Depthwise separable convolutions factorize a standard convolution into per-channel and cross-channel stages [40]. Quantization to 8-bit integers cuts model memory by 75% relative to 32-bit floating point and reduces memory footprint and latency on edge hardware [41]. On the uplink, FedPAQ established convergence guarantees for periodic averaging with quantized updates [42], and Stich et al. showed that retaining sparsification residuals in local memory preserves convergence [43].

Table 1 summarizes representative prior work. Fed-EdgeIDS combines depthwise separable convolutions [40], attention with matched tensor dimensions, error-compensated 8-bit stochastic quantization, trimmed-mean aggregation [44], server momentum [45], and Gaussian update perturbation with Rényi accounting [46, 47].

Table 1: Representative Prior Work Related to Fed-EdgeIDS

Study	Setting	Model	Main Emphasis
Ferrag et al. [8]	Centralized	Several deep models	Comparative evaluation of deep learning IDS on public datasets
Diro and Chilamkurti [12]	Distributed (fog)	Deep network	Distributed versus centralized attack detection
Tai et al. [23]	Centralized	RF, GBM, neural networks	Anomaly detection in industrial control systems
AbdelSatar et al. [25]	Centralized	Ensemble learners	Man-in-the-middle detection on TON_IoT
Wu et al. [34]	Centralized	Transformer	Self-attention for intrusion detection on CICIDS2017 and CIC-DDoS2019
Rey et al. [36]	Federated	MLP, autoencoder	Federated malware detection on N-BaIoT with poisoning and aggregation analysis
Mothukuri et al. [37]	Federated	GRU	Federated anomaly detection for IoT attacks
Nishio and Yonetani [38]	Federated	General	Client selection under resource heterogeneity
Wang et al. [39]	Federated	General	Adaptive aggregation frequency under a resource budget at the edge
Reisizadeh et al. [42]	Federated	General	Periodic averaging with quantized updates
Yin et al. [44]	Distributed	General	Byzantine-robust statistical aggregation

3. PROPOSED ARCHITECTURE AND MATHEMATICAL FOUNDATIONS

3.1 Three-Tier Operational Architecture

Figure 1 illustrates the Fed-EdgeIDS architecture for an industrial plant. Tier 1, the fieldbus perception layer, contains sensors, PLCs, drives, and actuators that communicate over Modbus TCP, EtherCAT, PROFINET, and industrial MQTT in control loops shorter than 10 ms. Tier 2 consists of ruggedized edge gateways attached to field Ethernet switches through passive SPAN mirror ports or optical network TAPs. Because traffic is ingested passively, packet inspection runs out of band and adds no switching delay to the PLC loop. Each gateway extracts flow features, maintains a ring buffer of training samples, and performs real-time inference with DSA-Net. A gateway either contains a detected attack out of band, for example by forging TCP RST packets for a misbehaving Modbus or MQTT session or by pushing blocking access control lists to a managed switch, or remains passive and only logs the event. Each gateway trains locally by minimizing the focal-loss and proximal objective of **Section 3.5**, clips and perturbs the resulting update with Gaussian noise, sparsifies it by top-k selection, and quantizes it to 8 bits before upload. Tier 3 is an on-premise central orchestration server at the enterprise SCADA level of the plant. It dequantizes and aggregates the incoming updates, applies the server momentum step, and broadcasts new global checkpoints to the gateways without ever accessing raw telemetry.

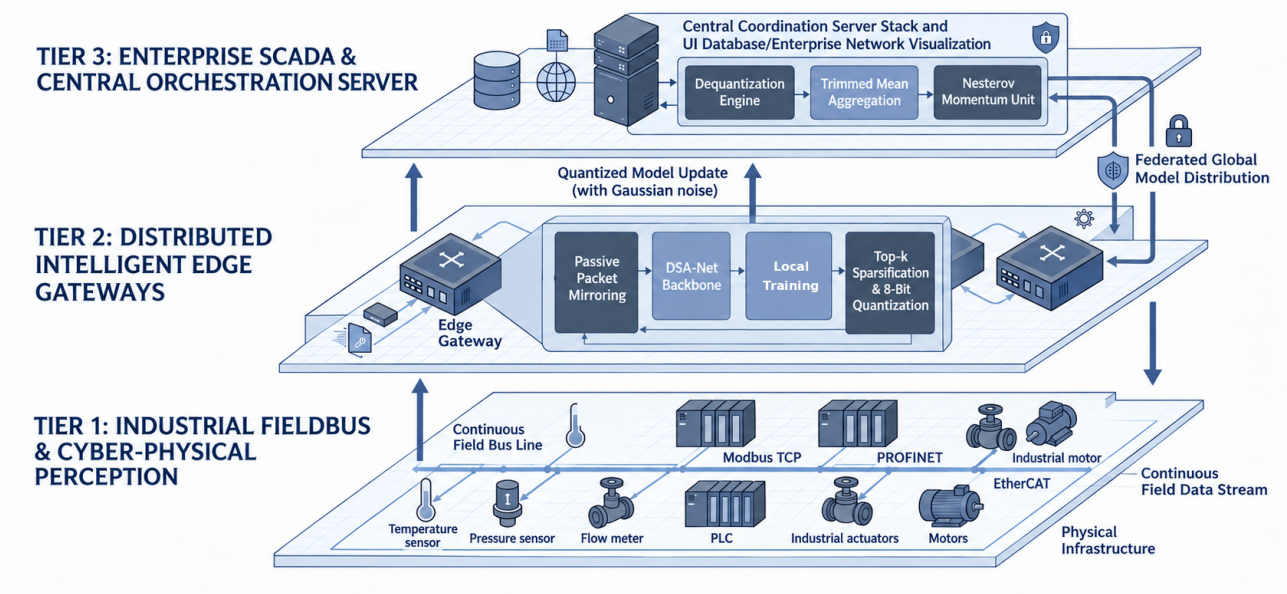

Figure 1: Multi-Tier Federated Edge AI Architecture of Fed-EdgeIDS

3.2 Threat Model

Two threat vectors are considered. Threat Vector A covers an external or insider attacker who injects malicious traffic into the plant subnet. The traffic families considered are denial-of-service and distributed denial-of-service flooding, information gathering by scanning and fingerprinting, man-in-the-middle spoofing, injection attacks, and malware including backdoors, password cracking, and ransomware. The Edge-IIoTset benchmark of **Section 4.1** realizes these families.

Threat Vector B targets the training loop. The orchestration server is modeled as honest-but-curious. It executes aggregation faithfully, but an insider or a compromised process on the server may attempt gradient inversion [19] on incoming updates to reconstruct private process telemetry. In addition, a subset of gateways may be compromised and upload poisoned updates through sign flipping ($\Delta w_{\text{byz}} = -c\,\Delta w_k$), high-variance Gaussian noise ($\mathcal{N}(0, 100\mathbf{I})$), or local label flipping. The trimmed-mean aggregator of **Section 3.8** is included as a defense against such updates, but its effect was not evaluated experimentally in this study (see **Section 5.4**).

3.3 Depthwise Separable One-Dimensional Convolutions

A standard one-dimensional convolution filters the sequence across all input channels jointly and therefore requires many multiply-accumulate (MAC) operations on a low-power CPU. Let the input window be $X \in \mathbb{R}^{L \times C_{\text{in}}}$, with window length $L = 10$ and $C_{\text{in}} = 45$ channels for Edge-IIoTset. DSA-Net factorizes each convolution into two steps following the depthwise separable design of [40]. As shown in **Equation 1**, the depthwise step filters each channel c independently with a kernel of

length S:

$$Y_d(t,c) = \sum_{s=0}^{S-1} W_d(s,c)\, X(t+s,c), \tag{1}$$

where $W_d \in \mathbb{R}^{S\times C_{\text{in}}}$ and the step costs $L\,S\,C_{\text{in}}$ MACs. As shown in **Equation 2**, the pointwise step then mixes channels with a 1×1 convolution:

$$Y_p(t,m) = \sum_{c=1}^{C_{\text{in}}} W_p(c,m)\, Y_d(t,c) + b(m), \tag{2}$$

where $W_p \in \mathbb{R}^{C_{\text{in}}\times C_{\text{out}}}$ and $b \in \mathbb{R}^{C_{\text{out}}}$, at a cost of $L\,C_{\text{in}}\,C_{\text{out}}$ MACs. A standard convolution costs $L\,S\,C_{\text{in}}\,C_{\text{out}}$ MACs, so the cost ratio of **Equations 1 and 2** is as shown in **Equation 3**:

$$\text{Ratio} = \frac{L\,S\,C_{\text{in}} + L\,C_{\text{in}}\,C_{\text{out}}}{L\,S\,C_{\text{in}}\,C_{\text{out}}} = \frac{1}{C_{\text{out}}} + \frac{1}{S}. \tag{3}$$

For the second stage of DSA-Net ($S = 5$, $C_{\text{out}} = 128$), **Equation 3** gives $1/128 + 1/5 = 0.2078$, a 79.2% reduction in MACs for that layer. Across both stages (channel widths of 45, 64, and 128), the factorized extractor needs $L(225 + 2880 + 320 + 8192) = 11{,}617\,L$ MACs against $55{,}360\,L$ for standard convolutions, a ratio of 0.2098 and a 79.0% reduction. Normalization and activation costs are excluded from this count.

3.4 Scaled Multi-Head Self-Attention and Layer Specification

Small kernels ($S = 5$) capture relationships between adjacent features but miss dependencies between protocol fields that are far apart in the window. The output of the second convolutional stage, $Z_{\text{conv}} \in \mathbb{R}^{L\times 128}$, is therefore passed to a multi-head self-attention module [33]. A linear projection $W_{\text{proj}} \in \mathbb{R}^{128\times d_{\text{model}}}$ first maps Z_{conv} to $Z \in \mathbb{R}^{L\times d_{\text{model}}}$ with $d_{\text{model}} = 64$. For each of the $h = 4$ heads, with $d_k = d_{\text{model}}/h = 16$, learned matrices $W_Q^{(i)}, W_K^{(i)}, W_V^{(i)} \in \mathbb{R}^{64\times 16}$ produce the query, key, and value tensors $\mathbf{Q}_i$, $\mathbf{K}_i$, and $\mathbf{V}_i$. As shown in **Equation 4**, each head computes scaled dot-product attention:

$$\text{head}_i = \text{softmax}\!\left(\frac{\mathbf{Q}_i\mathbf{K}_i^{\top}}{\sqrt{d_k}}\right)\mathbf{V}_i. \tag{4}$$

The four head outputs are concatenated and projected back to 128 channels by $W_O \in \mathbb{R}^{64\times 128}$, which permits an exact residual connection with Z_{conv}. As shown in **Equation 5**, the result is normalized with layer normalization:

$$Z_{\text{out}} = \text{LayerNorm}\big(Z_{\text{conv}} + \text{Concat}(\text{head}_1,\ldots,\text{head}_4)\,W_O\big)\,. \tag{5}$$

Global average pooling then collapses the sequence dimension and dense layers produce the C-class softmax output. **Table 2** shows the layer configuration. Parameter counts include biases and the affine parameters of batch normalization, and global average pooling contributes no trainable parameters. The sum of all rows is 67,831 trainable parameters, a 94.6% reduction relative to the 1,248,500-parameter baseline M1. The attention-free backbone M2 omits the bottleneck projection, the attention block, the output projection, and the layer normalization (29,312 parameters in total) and therefore has 38,519 parameters, a 96.9% reduction.

3.5 Local Optimization with Proximal Regularization and Focal Loss

Routine polling frames greatly outnumber attack packets, and class ratios differ across gateways. Gateway k therefore trains its local weights w_k on its private buffer $\mathcal{D}_k = \{(x_i, y_i)\}_{i=1}^{n_k}$ by minimizing the sum of a focal loss and a FedProx-style proximal term [48], as shown in **Equation 6**:

$$F_k(w_k) = \frac{1}{n_k}\sum_{i=1}^{n_k} \mathcal{L}_{\text{FL}}\big(f(x_i;w_k), y_i\big) + \frac{\mu}{2}\,\|w_k - w^{(t)}\|_2^2. \tag{6}$$

As shown in **Equation 7**, the focal loss [49] down-weights easy samples so that minority attack classes contribute more to the gradient:

$$\mathcal{L}_{\text{FL}}(p_t) = -\alpha_t\,(1-p_t)^{\gamma}\log(p_t), \tag{7}$$

with $\gamma = 2.0$ and $\alpha_t = 0.25$. Since α_t is a constant scalar, class emphasis arises from the modulating factor $(1-p_t)^{\gamma}$ and not from per-class weights. The proximal weight $\mu = 0.01$ limits drift from the broadcast weights $w^{(t)}$ over $E = 3$ local epochs, trained with the Adam optimizer [50] at a learning rate of 0.001 and a batch size of 64.

Table 2: Layer Configuration and Parameter Breakdown of DSA-Net

Stage	Operation	Input	Output	Parameters
Input	Normalized window ($L = 10$, $C_{\text{in}} = 45$)	$(B, 10, 45)$	$(B, 10, 45)$	0
Conv stage 1	Depthwise conv ($S = 5$, stride 1, pad 2), BN, ReLU	$(B, 10, 45)$	$(B, 10, 45)$	360
	Pointwise conv (1×1, 64 filters), BN, ReLU	$(B, 10, 45)$	$(B, 10, 64)$	3,072
Conv stage 2	Depthwise conv ($S = 5$, stride 1, pad 2), BN, ReLU	$(B, 10, 64)$	$(B, 10, 64)$	512
	Pointwise conv (1×1, 128 filters), BN, ReLU (Z_{conv})	$(B, 10, 64)$	$(B, 10, 128)$	8,576
Bottleneck	Linear projection W_{proj} ($128 \rightarrow 64$)	$(B, 10, 128)$	$(B, 10, 64)$	8,256
Attention	Four-head self-attention ($d_{\text{model}} = 64$, $d_k = 16$)	$(B, 10, 64)$	$(B, 10, 64)$	12,480
Output projection	Linear projection W_O ($64 \rightarrow 128$)	$(B, 10, 64)$	$(B, 10, 128)$	8,320
Residual and norm	Residual addition and LayerNorm	$(B, 10, 128)$	$(B, 10, 128)$	256
Pooling	Global average pooling	$(B, 10, 128)$	$(B, 128)$	0
Classifier	Dense 128, BN, ReLU, dense 64	$(B, 128)$	$(B, 64)$	25,024
Output	Dense $64 \rightarrow C = 15$, softmax	$(B, 64)$	$(B, 15)$	975
Total	**DSA-Net (models M3 to M5)**			**67,831**

3.6 Gaussian Update Perturbation and Privacy Accounting

An honest-but-curious server could attempt gradient inversion on the update $\Delta w_k = w_k - w^{(t)}$. Each gateway therefore perturbs its update before upload, following the clipping and Gaussian noise construction used for differentially private deep learning [46, 51]. As shown in **Equation 8**, the update is first scaled to an L_2 norm of at most $C_{\text{clip}} = 1.0$:

$$\bar{g}_k = \frac{\Delta w_k}{\max\left(1,\ \|\Delta w_k\|_2 / C_{\text{clip}}\right)}. \tag{8}$$

As shown in **Equation 9**, zero-mean Gaussian noise with noise multiplier $\sigma = 0.85$ is then added:

$$\tilde{g}_k = \bar{g}_k + \mathcal{N}\left(\mathbf{0},\ \sigma^2 C_{\text{clip}}^2\, \mathbf{I}\right). \tag{9}$$

The privacy loss of this mechanism is quantified with Rényi differential privacy [47]. The server observes each noisy update individually and knows which gateways participate, so sampling provides no amplification. For the presence or absence of one clipped update, a single release is a Gaussian mechanism with sensitivity C_{clip} and therefore satisfies $(\alpha, \alpha/(2\sigma^2))$-Rényi differential privacy for every order $\alpha > 1$. A gateway that participates in n_p rounds composes n_p such releases, which gives $R(\alpha) = n_p \alpha/(2\sigma^2)$. Converting to (ε, δ) with $\varepsilon = R(\alpha) + \ln(1/\delta)/(\alpha - 1)$ [47] and minimizing over α yields the closed form shown in **Equation 10**:

$$\varepsilon = c + 2\sqrt{c\ \ln(1/\delta)}, \qquad c = \frac{n_p}{2\sigma^2}. \tag{10}$$

With client sampling ratio $q = m/K = 4/20 = 0.20$ and $T = 50$ rounds, a gateway participates in $n_p = qT = 10$ rounds on average, and the bound is evaluated at this value. For $\sigma = 0.85$ and $\delta = 10^{-5}$, **Equation 10** gives $\varepsilon \approx 24.8$ at the client level. This bound is weak in the formal sense. The perturbation therefore acts as obfuscation of individual updates against gradient inversion, in the spirit of the noise defenses studied in [19], and not as a strong formal privacy guarantee. Its protective effect against gradient inversion is not measured in this study. **Section 5.3** relates the noise multiplier to the cumulative bound and to accuracy.

3.7 Stochastic 8-bit Quantization and Sparsification

Uploading dense 32-bit vectors every round loads constrained plant links. Each gateway keeps a residual buffer e_k, initialized to zero. As shown in **Equation 11**, the residual is added to the perturbed update before compression:

$$p_k^{(t)} = \tilde{g}_k + e_k^{(t)}. \tag{11}$$

Magnitude-based top-k sparsification keeps the index set Ω_k holding the largest 30% of coordinates ($\rho = 0.30$, which retains 20,349 of 67,831 coordinates) and zeroes the remaining 70%. With the dynamic scale $s_k = \max_{j\in\Omega_k} |p_k^{(t)}(j)|/127$, each retained coordinate v is mapped to a signed 8-bit integer by the stochastic rounding operator shown in **Equation 12**, in the spirit of gradient quantization [52]:

$$Q_8(v, s_k) = s_k \ \mathrm{sign}(v) \left\lfloor \frac{|v|}{s_k} + \xi \right\rfloor, \qquad \xi \sim \mathrm{Uniform}(0, 1). \tag{12}$$

The rounding operator is conditionally unbiased, $\mathbb{E}_\xi[Q_8(v, s_k)] = v$, whereas deterministic top-k truncation is biased. As shown in **Equation 13**, the discarded coordinates and the quantization error are stored in the residual buffer, which compensates the sparsification bias across rounds and preserves convergence [43]:

$$e_k^{(t+1)} = p_k^{(t)} - \hat{g}_k^{(t)}. \tag{13}$$

Here $\hat{g}_k^{(t)}$ denotes the sparse quantized update. Because sparsification, quantization, and the residual buffer all act on the already perturbed update, these steps are post-processing and leave the bound of **Section 3.6** unchanged.

Communication accounting: With $m = 4$ of $K = 20$ gateways sampled per round, 200 client uploads occur over 50 rounds. Sizes are given in bytes and in MB with 1 MB = 2^{20} bytes. The dense 32-bit baseline model M1 occupies 4,994,000 bytes (4.76 MB) per upload, and dense 32-bit DSA-Net occupies 271,324 bytes (0.259 MB), a 94.6% reduction. A sparse packet carries 20,349 one-byte values, 20,349 indices stored as 16-bit gaps (40,698 bytes), and a four-byte scale, for a total of 61,051 bytes (0.058 MB) per upload. This is 77.5% below the dense DSA-Net upload and 98.8% below the baseline upload. Of the 77.5% saving, 75.0 points stem from 8-bit quantization, and sparsification reduces a dense 8-bit upload of 67,835 bytes by a further 10.0%. Over 200 uploads, the cumulative payload is $200 \times 61{,}051 = 12{,}210{,}200$ bytes (11.6 MB), against 998,800,000 bytes (952.5 MB) for uncompressed federated averaging of the baseline. All counts refer to application payload and exclude transport framing. The server broadcast is not compressed in this study, so 200 dense downloads of DSA-Net add 54,264,800 bytes (51.7 MB). On the 10 Mbps link of the testbed, the payload of one sparse upload takes about 0.05 s to transmit, whereas a dense baseline upload takes about 4.0 s, excluding delay and loss. **Table 3** lists the per-upload and cumulative payloads.

Table 3: Uplink Payload per Upload and over 50 Rounds (200 Uploads)

Configuration	Bytes per upload	MB per upload	Total over 50 rounds (MB)	Time per upload at 10 Mbps (s)
Baseline 1D-CNN (M1), dense 32-bit	4,994,000	4.763	952.5	4.00
DSA-Net (M3), dense 32-bit	271,324	0.259	51.8	0.22
Fed-EdgeIDS (M4, M5), sparse 8-bit	**61,051**	**0.058**	**11.6**	**0.05**

3.8 Trimmed-Mean Aggregation and Server Momentum

Plain averaging lets a single gateway that sends large negative weights corrupt the global model. After dequantization, the server instead sorts the n_t received updates independently at each coordinate j, so that $\Delta w_{(1),j} \le \cdots \le \Delta w_{(n_t),j}$. It discards the $m_{\mathrm{trim}} = \lfloor \beta\, n_t \rfloor$ largest and m_{trim} smallest values and averages the rest, as shown in **Equation 14**:

$$\Delta w_j^{(t)} = \frac{1}{n_t - 2m_{\mathrm{trim}}} \sum_{r=m_{\mathrm{trim}}+1}^{n_t - m_{\mathrm{trim}}} \Delta w_{(r),j}. \tag{14}$$

The trimming budget depends on the number of received updates. In the main training runs, $n_t = m = 4$ and $\lfloor 0.20 \times 4 \rfloor = 0$, so the aggregator reduces to the mean of the four updates. Trimming becomes active only when at least five updates are aggregated.

The analysis in [44] shows that coordinate-wise trimmed mean tolerates a fraction α of corrupted workers when $\beta \ge \alpha$, but under assumptions on the gradient distribution and on homogeneous data across workers. The non-IID, noisy, and quantized updates used here do not satisfy these assumptions, so $\beta = 0.20$ serves as a nominal design parameter. Robustness of Fed-EdgeIDS to Byzantine gateways was not evaluated in the experiments reported here.

The server then updates the global weights with Nesterov momentum [53], in the form implemented by common deep learning libraries ($\gamma_m = 0.9$, server step $\eta_{\mathrm{server}} = 1.0$), which complements server-side adaptive optimization in federated learning [45]. As shown in **Equations 15 and 16**, the velocity and weights are updated as

$$v^{(t+1)} = \gamma_m\, v^{(t)} + \Delta w^{(t)}, \tag{15}$$

$$w^{(t+1)} = w^{(t)} + \eta_{\mathrm{server}} \left(\Delta w^{(t)} + \gamma_m\, v^{(t+1)} \right). \tag{16}$$

Figure 2 shows one communication round, and **Algorithm 1** presents the complete procedure.

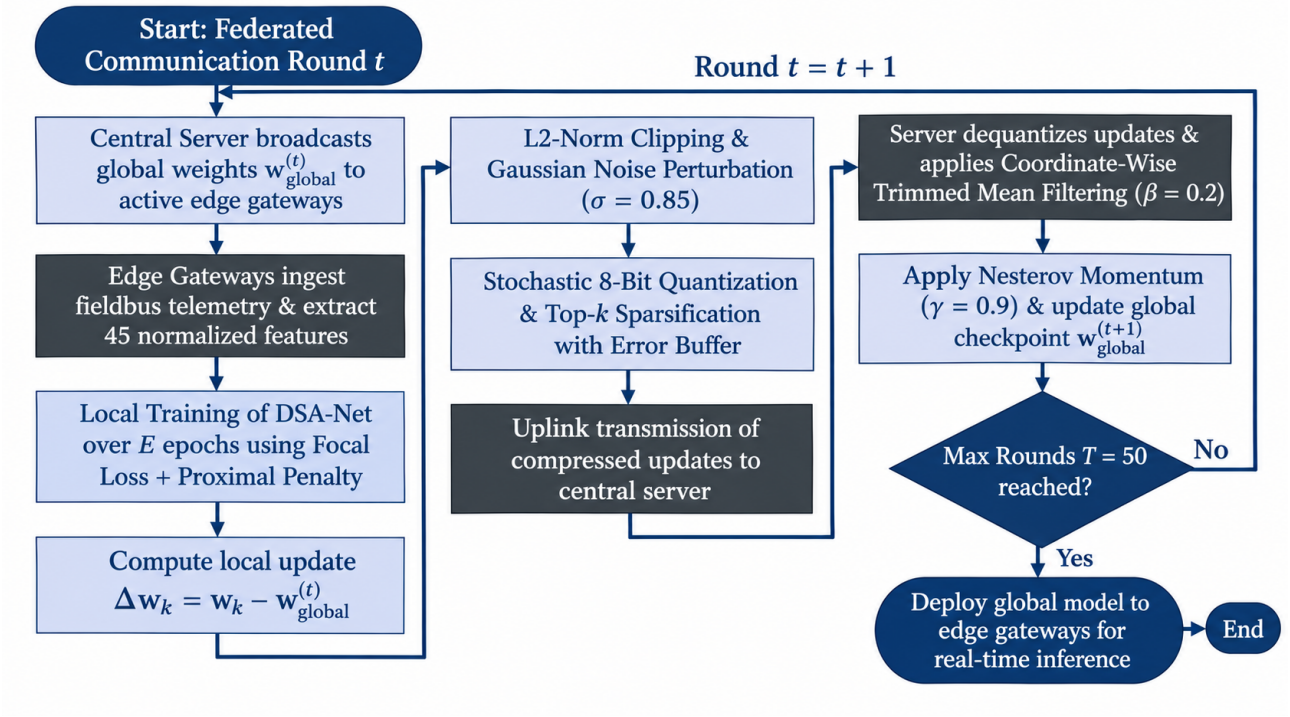

Figure 2: Round Lifecycle of Local Optimization, Perturbation, Quantization, and Trimmed-Mean Aggregation

Algorithm 1 Quantized and Perturbed Federated Edge Optimization

Require: Gateway set $\mathcal{K}$ with $K = 20$; $m = 4$ gateways per round; $T = 50$ rounds; $E = 3$ local epochs; batch size 64; learning rate 0.001; $\mu = 0.01$; $C_{\text{clip}} = 1.0$; $\sigma = 0.85$; $\rho = 0.30$; $\beta = 0.20$; $\gamma_m = 0.9$; $\eta_{\text{server}} = 1.0$

Ensure: Global model $w^{(T)}$

Initialize $w^{(0)}$, $v^{(0)} \leftarrow \mathbf{0}$, and $e_k^{(0)} \leftarrow \mathbf{0}$ for all $k \in \mathcal{K}$
for $t = 0, 1, \ldots, T-1$ **do**
  Sample $\mathcal{S}_t \subset \mathcal{K}$ with $|\mathcal{S}_t| = m$ and broadcast $w^{(t)}$
  for each gateway $k \in \mathcal{S}_t$ in parallel **do**
    $w_k \leftarrow w^{(t)}$; minimize F_k of **Equation 6** over $\mathcal{D}_k$ for E epochs
    $\Delta w_k \leftarrow w_k - w^{(t)}$
    $\bar{g}_k \leftarrow \Delta w_k / \max(1, \|\Delta w_k\|_2 / C_{\text{clip}})$ ▷ **Equation 8**
    $\tilde{g}_k \leftarrow \bar{g}_k + \mathcal{N}(\mathbf{0}, \sigma^2 C_{\text{clip}}^2 \mathbf{I})$ ▷ **Equation 9**
    $p_k \leftarrow \tilde{g}_k + e_k$; select Ω_k as the top ρ fraction of $|p_k|$ ▷ **Equation 11**
    $s_k \leftarrow \max_{j \in \Omega_k} |p_k(j)|/127$; $\hat{g}_k(j) \leftarrow Q_8(p_k(j), s_k)$ for $j \in \Omega_k$, 0 otherwise ▷ **Equation 12**
    $e_k \leftarrow p_k - \hat{g}_k$; transmit $(\hat{g}_k|_{\Omega_k}, \Omega_k, s_k)$ ▷ **Equation 13**
  end for
  Dequantize received updates and compute $\Delta w^{(t)}$ by trimmed mean ▷ **Equation 14**
  $v^{(t+1)} \leftarrow \gamma_m v^{(t)} + \Delta w^{(t)}$; $w^{(t+1)} \leftarrow w^{(t)} + \eta_{\text{server}}(\Delta w^{(t)} + \gamma_m v^{(t+1)})$ ▷ **Equations 15 and 16**
end for
return $w^{(T)}$

4. EXPERIMENTAL EVALUATION

4.1 Benchmark Datasets and Preprocessing

Two public benchmarks are used, and **Table 4** summarizes the properties of both.

Edge-IIoTset: Edge-IIoTset [54] was generated on a purpose-built IoT and IIoT testbed with more than ten device types, including Modbus TCP sensor traffic. The release describes its records by 61 flow features that were selected from 1,176 candidates. Fourteen attack types fall into five threat categories: denial of service and distributed denial of service (TCP

SYN, UDP, HTTP, and ICMP floods), information gathering (port scanning, OS fingerprinting, and vulnerability scanning), man-in-the-middle, injection (cross-site scripting, SQL injection, and file uploading), and malware (backdoor, password cracking, and ransomware). Together with normal traffic, these form 15 classes. Class sizes differ strongly across the attack types, which motivates the focal loss of **Equation 7**. After removal of constant and identifier columns, 45 input features remain.

CIC-IDS2017: CIC-IDS2017 [55] contains 2,830,743 labeled network flows with 78 statistical flow features, recorded over five days of benign background traffic and scripted attacks on an enterprise network. Its attack labels cover several DoS variants, Heartbleed, DDoS, port scanning, botnet activity, FTP and SSH brute force, three web attack types, and infiltration. The labels are merged here into benign traffic and seven families (DoS, DDoS, PortScan, Botnet, Infiltration, Web Attack, and Patator brute force), which gives 8 classes. CIC-IDS2017 is an enterprise IT benchmark. It contains no Modbus traffic, no Modbus-specific attacks, and no false data injection, and its infiltration label is a single class of infiltration activity. The benchmark therefore tests generalization to conventional network attacks and supports no claim about industrial protocol attacks. Results on both datasets are reported as aggregate metrics, and no per-class analysis is given.

Table 4: Dataset Properties and Experimental Parameters

Property	Edge-IIoTset [54]	CIC-IDS2017 [55]
Target domain	IoT and IIoT testbed with Modbus TCP traffic	Enterprise IT network
Records	Selected machine learning subset of the public release	2,830,743 flows
Input features	61 released; 45 after preprocessing	78 statistical flow features
Classes evaluated	15 (normal and 14 attacks)	8 (benign and 7 attack families)
Attack groups	DoS and DDoS, information gathering, man-in-the-middle, injection, malware	DoS, DDoS, PortScan, Botnet, Infiltration, Web Attack, Patator brute force
Industrial protocol attacks	Not protocol-specific; Modbus TCP is present as device traffic	None
Partitioning	70/15/15 split; non-IID Dirichlet ($\alpha = 0.5$)	70/15/15 split; non-IID Dirichlet ($\alpha = 0.5$)
Gateways	$K = 20$, $m = 4$ sampled per round	$K = 20$, $m = 4$ sampled per round

Flows are grouped into sessions before the split to avoid temporal leakage, and the data are divided into 70% training, 15% validation, and 15% held-out test sets. Constant and identifier columns are removed, and min-max scaling fitted on the training set is applied to all splits. Consecutive flows are then stacked into windows of $L = 10$. Results are reported as the mean ± standard deviation over five random seeds (42, 123, 456, 789, and 999).

4.2 Experimental Testbed

Experiments ran on a workstation with an Intel Xeon E5-2690 v4 CPU hosting 20 gateway processes. Different factory zones observe different traffic mixes, so training samples were distributed over the 20 gateways with a Dirichlet distribution ($\alpha = 0.5$), which gives each gateway a skewed class mixture. The Linux `tc-netem` utility limited the uplink to 10 Mbps with a 15 ms round-trip delay and 0.5% packet loss. On-device profiling used a Raspberry Pi 4 Model B and an NVIDIA Jetson Nano. The data split ratios, random seeds, Dirichlet parameter, network emulation settings, and all training and compression hyperparameters are listed in this section and in **Algorithm 1**, and the simulation scripts are available from the corresponding author on reasonable request.

4.3 Baseline Models

Five models were trained on identical splits and seeds. The first is a centralized CNN-LSTM that pools the raw flows of all 20 gateways on one server and combines the convolutional and recurrent model families compared in [8]. The second is standard FedAvg [15] with a three-layer one-dimensional CNN and uncompressed 32-bit uploads. The third, Fed-MLP [36], uses a fully connected network. The fourth, Fed-GRU [37], uses a gated recurrent network. The fifth, FedPAQ [42], uses a one-dimensional CNN with periodic averaging and quantized updates, and omits attention, proximal regularization, and trimmed-mean aggregation.

4.4 Classification Performance and Communication Overhead

Table 5 lists accuracy, precision, recall, F1-score, and false alarm rate (FAR). The FAR is the fraction of normal flows classified as any attack class, and all metrics are evaluated once on the held-out test split after round 50. On Edge-IIoTset, Fed-EdgeIDS reaches 98.74% ± 0.12% accuracy and a 98.61% ± 0.14% F1-score, which is 0.31 percentage points below the centralized CNN-LSTM (99.05% ± 0.08%) without exporting raw telemetry. Its FAR of 0.62% ± 0.05% compares with 1.74% ± 0.12% for standard FedAvg and 2.10% ± 0.14% for FedPAQ. On CIC-IDS2017, Fed-EdgeIDS reaches 99.02% ± 0.09% accuracy, a 98.91% ± 0.11% F1-score, and a FAR of 0.49% ± 0.04%, which is 0.19 percentage points below the centralized CNN-LSTM (99.21% ± 0.06%) and ahead of Fed-MLP (94.62%), FedPAQ (96.35%), standard FedAvg (97.10%), and Fed-GRU (97.45%).

Table 5: Classification Performance (Mean ± Std over Five Runs)

Framework	Dataset	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	FAR (%)
Centralized CNN-LSTM	Edge-IIoTset	99.05 ± 0.08	98.92 ± 0.10	99.01 ± 0.09	98.96 ± 0.09	0.48 ± 0.04
Standard FedAvg [15]	Edge-IIoTset	96.42 ± 0.28	96.15 ± 0.31	96.38 ± 0.29	96.26 ± 0.30	1.74 ± 0.12
Fed-MLP [36]	Edge-IIoTset	94.18 ± 0.35	93.85 ± 0.38	94.02 ± 0.36	93.93 ± 0.37	2.89 ± 0.18
Fed-GRU [37]	Edge-IIoTset	96.88 ± 0.22	96.70 ± 0.25	96.81 ± 0.23	96.75 ± 0.24	1.42 ± 0.09
FedPAQ [42]	Edge-IIoTset	95.80 ± 0.31	95.44 ± 0.34	95.72 ± 0.32	95.58 ± 0.33	2.10 ± 0.14
Fed-EdgeIDS	Edge-IIoTset	**98.74 ± 0.12**	**98.65 ± 0.13**	**98.58 ± 0.14**	**98.61 ± 0.14**	**0.62 ± 0.05**
Centralized CNN-LSTM	CIC-IDS2017	99.21 ± 0.06	99.10 ± 0.08	99.18 ± 0.07	99.14 ± 0.07	0.39 ± 0.03
Standard FedAvg [15]	CIC-IDS2017	97.10 ± 0.24	96.84 ± 0.26	97.05 ± 0.25	96.94 ± 0.25	1.35 ± 0.10
Fed-MLP [36]	CIC-IDS2017	94.62 ± 0.32	94.20 ± 0.35	94.55 ± 0.33	94.37 ± 0.34	2.45 ± 0.16
Fed-GRU [37]	CIC-IDS2017	97.45 ± 0.19	97.18 ± 0.22	97.38 ± 0.20	97.28 ± 0.21	1.18 ± 0.08
FedPAQ [42]	CIC-IDS2017	96.35 ± 0.27	96.02 ± 0.30	96.28 ± 0.28	96.15 ± 0.29	1.82 ± 0.12
Fed-EdgeIDS	CIC-IDS2017	**99.02 ± 0.09**	**98.94 ± 0.11**	**98.88 ± 0.10**	**98.91 ± 0.11**	**0.49 ± 0.04**

Communication: **Table 3** introduces the uplink payload. With 200 uploads over 50 rounds, uncompressed federated averaging of the one-dimensional CNN transmits 952.5 MB, whereas Fed-EdgeIDS transmits 11.6 MB, a reduction of 98.8%. Two factors contribute: the compact DSA-Net reduces the dense upload by 94.6%, and the sparse 8-bit packet is a further 77.5% smaller than the dense 32-bit DSA-Net upload. The payloads of the remaining baselines depend on model sizes that are not specified in this study and are therefore not reported. On the emulated 10 Mbps link, the payload of one baseline upload takes 4.0 s to transmit against 0.05 s for Fed-EdgeIDS.

5. DISCUSSION

5.1 Ablation of Architectural Components

Table 6 reports seven configurations on Edge-IIoTset. The baseline one-dimensional CNN with uncompressed FedAvg (M1) has 1,248,500 parameters, reaches 96.42% ± 0.28% accuracy, uploads 4.76 MB per client round, and needs 8.92 ms per window on the Raspberry Pi 4. Replacing standard convolutions with depthwise separable ones (M2) removes 96.9% of the parameters and cuts latency by 76.0% to 2.14 ms, but accuracy falls by 0.54 percentage points to 95.88%.

Adding the bottleneck projection and the four-head attention block (M3, complete DSA-Net) increases the parameter count by 29,312 to 67,831, which remains a 94.6% reduction relative to M1, and raises accuracy to 98.81% and the F1-score by 2.96 points (from 95.74% to 98.70%). Latency of the 32-bit model rises to 3.94 ms. Sparsification and 8-bit quantization with error feedback (M4) shrink the upload from 0.259 MB to 0.058 MB at a cost of 0.03 points of accuracy (98.81%

to 98.78%). To separate perturbation from aggregation, M5a adds Gaussian update perturbation ($\sigma = 0.85$) under plain FedAvg, which lowers accuracy by 0.42 points to 98.36% ± 0.18%. M5b applies the trimmed-mean aggregator and Nesterov momentum without noise and reaches 98.85% ± 0.10%. Because only four updates are aggregated per round, trimming is inactive in these runs (**Section 3.8**), so the difference between M5a and M5 isolates the effect of server momentum. The complete system (M5) reaches 98.74% ± 0.12% accuracy and a 98.61% ± 0.14% F1-score, which recovers 0.38 points relative to M5a, and the perturbation costs 0.11 points relative to M5b. Differences below roughly 0.2 points between adjacent configurations are comparable to the seed standard deviations (0.09 to 0.18 points) and are not interpreted as separate effects.

Table 6: Ablation of Fed-EdgeIDS Components on Edge-IIoTset

Configuration	Parameters	Accuracy (%)	F1 (%)	Upload (MB)	Latency (ms)
M1: 1D-CNN with FedAvg	1,248,500	96.42 ± 0.28	96.26 ± 0.30	4.763	8.92 ± 0.18
M2: Depthwise separable, no attention	38,519	95.88 ± 0.32	95.74 ± 0.33	0.147	2.14 ± 0.05
M3: DSA-Net (M2 with attention)	67,831	98.81 ± 0.11	98.70 ± 0.12	0.259	3.94 ± 0.09
M4: M3 with 8-bit and top-k	67,831	98.78 ± 0.12	98.66 ± 0.13	0.058	3.74 ± 0.08
M5a: M4 with Gaussian noise, plain FedAvg	67,831	98.36 ± 0.18	98.22 ± 0.20	0.058	3.74 ± 0.08
M5b: M4 with trimmed mean and momentum, no noise	67,831	98.85 ± 0.10	98.74 ± 0.11	0.058	3.74 ± 0.08
M5: Complete Fed-EdgeIDS	**67,831**	**98.74 ± 0.12**	**98.61 ± 0.14**	**0.058**	**3.74 ± 0.08**

Note: Upload: dense 32-bit payload for M1 to M3 and sparse 8-bit payload for M4 to M5. Latency of M1 to M3 is for 32-bit models on the Raspberry Pi 4. M4 to M5 use the INT8 deployment checkpoint, and training-side components leave the inference graph unchanged.

5.2 Hardware Profiling on Embedded Devices

Table 7 lists measurements on the Raspberry Pi 4 and the Jetson Nano. The converged DSA-Net checkpoint was compiled with post-training INT8 quantization, using ONNX Runtime with ARM NEON instructions on the Raspberry Pi 4. On the Jetson Nano, the same checkpoint was deployed with TensorRT. This on-device quantization is separate from the uplink quantization of **Section 3.7**. To isolate numerical precision from architecture, the same DSA-Net was benchmarked on the Raspberry Pi at three precisions: FP32 (3.94 ± 0.09 ms, 182 MB), FP16 (3.82 ± 0.08 ms, 164 MB), and INT8 (3.74 ± 0.08 ms, 148 MB). Throughput is the reciprocal of the mean per-window latency. Architectural factorization from the baseline (8.92 ms) to DSA-Net FP32 (3.94 ms) accounts for most of the gain, a 55.8% latency reduction, and INT8 quantization adds a further 5.1% together with 18.7% lower memory. Relative to the baseline on the same board, INT8 DSA-Net uses 64.8% less RAM and delivers 2.4 times the throughput.

Table 7: Hardware Profiling, Precision Comparison, and End-to-End Latency

Platform	Model and precision	Latency (ms)	Throughput (windows/s)	RAM (MB)	Load (%)
Raspberry Pi 4	DSA-Net, FP32	3.94 ± 0.09	253.8	182	31.6 (CPU)
Raspberry Pi 4	DSA-Net, FP16	3.82 ± 0.08	261.8	164	29.8 (CPU)
Raspberry Pi 4	**DSA-Net, INT8 (proposed)**	**3.74 ± 0.08**	**267.4**	**148**	**28.4 (CPU)**
Jetson Nano	**DSA-Net, TensorRT**	**1.18 ± 0.03**	**847.5**	**312**	**18.2 (GPU)**
Raspberry Pi 4	Baseline M1, FP32	8.92 ± 0.18	112.1	420	74.8 (CPU)

Pipeline stage	Raspberry Pi 4 (ms)	Jetson Nano (ms)
1. Packet ingestion and ring-buffer capture	0.12 ± 0.01	0.08 ± 0.01
2. Feature extraction and normalization	0.48 ± 0.03	0.34 ± 0.02
3. DSA-Net inference (INT8 on Raspberry Pi 4)	3.74 ± 0.08	1.18 ± 0.03
4. Thresholding and containment dispatch	0.22 ± 0.02	0.22 ± 0.02
Total, detection to dispatch	**4.56 ± 0.10**	**1.82 ± 0.05**

The lower half of **Table 7** profiles the pipeline from packet capture to containment dispatch over a sliding window of the $L = 10$ most recent flows. Summing ring-buffer capture (0.12 ms), feature extraction (0.48 ms), inference (3.74 ms), and

containment dispatch (0.22 ms) gives 4.56 ± 0.10 ms on the Raspberry Pi 4. The corresponding Jetson Nano stages (0.08, 0.34, 1.18, and 0.22 ms) sum to 1.82 ± 0.05 ms. Both mean totals lie below the 10 ms control-loop target of **Section 3.1**. The totals are averages over repeated runs, not worst-case execution times, and exclude window accumulation, propagation of the containment command, and enforcement by the switch or controller. The results therefore support soft real-time operation and do not establish a hard deadline guarantee. **Figure 3** plots the stage latencies of the lower half of **Table 7**.

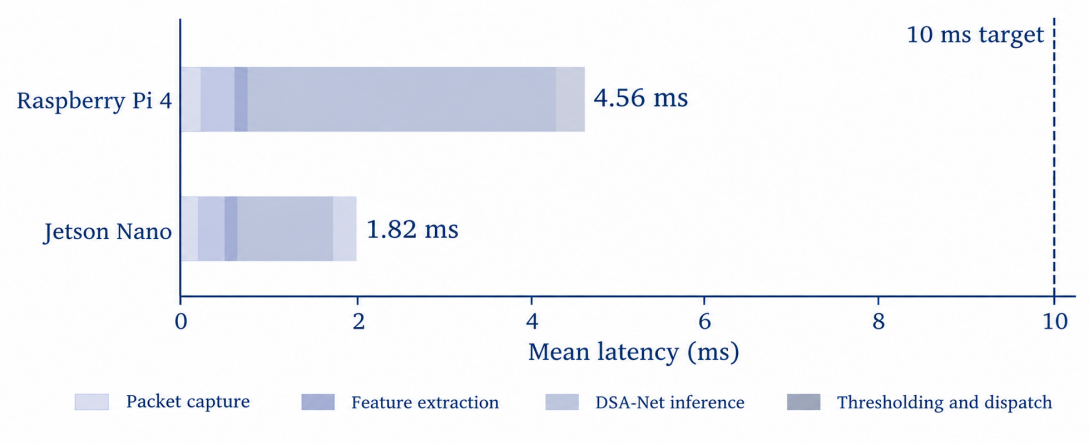

Figure 3: Stage Latencies from Packet Capture to Containment Dispatch on the Raspberry Pi 4 and the Jetson Nano

Figure 4 compares the three precisions of **Table 7** on the Raspberry Pi 4 in terms of dense uplink payload and latency. The payload follows directly from the bit width applied to the 67,831 parameters (0.259, 0.129, and 0.065 MB). Precisions below 8 bits were not evaluated for detection accuracy and are therefore not considered.

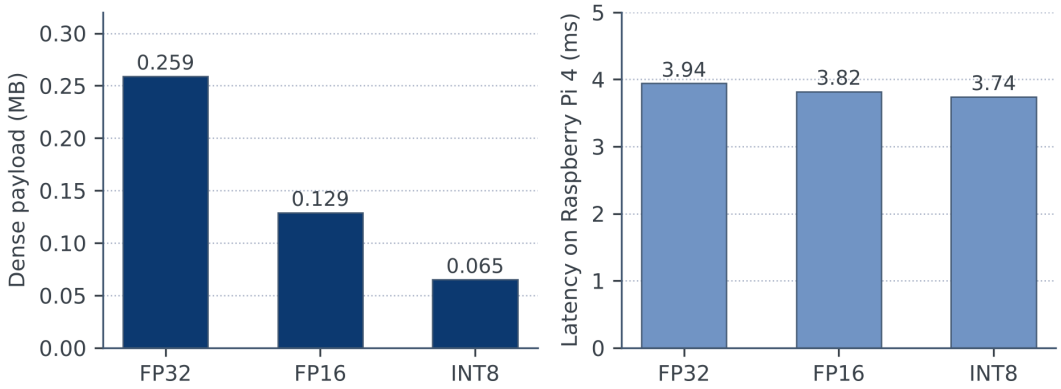

Figure 4: Dense Uplink Payload and Inference Latency of DSA-Net by Numerical Precision

5.3 Noise-Utility Trade-off

A sensitivity sweep varied the noise multiplier over $\sigma \in \{2.45, 1.38, 0.85, 0.56, 0.38, 0\}$ with all other settings fixed. **Table 8** lists the cumulative client-level bound of **Equation 10** at $\delta = 10^{-5}$ and $n_p = 10$, together with the measured accuracy. Accuracy at $\sigma = 0.85$ (98.74% ± 0.12%) is within 0.11 points of the noise-free model (98.85% ± 0.10%), and further reductions of σ add at most 0.08 points. Increasing σ to 1.38 and 2.45 lowers accuracy to 96.85% ± 0.22% and 91.24% ± 0.38%, while the bound remains between 7.0 and 13.6. Even the largest noise level of the sweep yields a bound above the values usually regarded as strong, so the perturbation is treated as obfuscation and not as a formal guarantee.

5.4 Deployment Considerations and Limitations

Two deployment practices follow from the architecture. First, because gateways exchange weights with the control-room server, the monitoring port should connect through a passive optical tap or a unidirectional hardware data diode, so that the monitoring link cannot inject packets onto the live PLC bus. Second, containment acts after detection, so the first malicious frames of an attack still reach the controller. The detector is therefore a complement to, and not a replacement for, protocol allow-listing and network segmentation.

Table 8: Noise Multiplier, Cumulative Privacy Bound, and Accuracy

Noise multiplier σ	**Coefficient $c = n_p/(2\sigma^2)$**	**Bound ε at $\delta = 10^{-5}$**	**Accuracy (%)**
2.45	0.83	7.0	91.24 ± 0.38
1.38	2.63	13.6	96.85 ± 0.22
0.85	**6.92**	**24.8**	**98.74 ± 0.12**
0.56	15.94	43.0	98.80 ± 0.10
0.38	34.63	74.6	98.82 ± 0.09
0	0	unbounded	98.85 ± 0.10

Several limitations apply. Federated training was simulated on a single workstation with emulated network impairments, and only inference and update computation were measured on physical boards. Both datasets are public benchmarks and not live plant captures. The privacy accounting is client-level and weak in the formal sense, and the protective effect of the noise against gradient inversion was not measured. With four updates per round, the trimmed mean reduces to the mean, so the robustness of the aggregator to Byzantine gateways was not evaluated. Per-class results and ROC curves are not reported, and the latency totals exclude window accumulation and enforcement time.

6. CONCLUSION

This paper presented Fed-EdgeIDS, a federated intrusion detection framework for resource-constrained industrial IoT gateways. The DSA-Net classifier, which combines depthwise separable convolutions with four-head self-attention, has 67,831 trainable parameters, a 94.6% reduction relative to a one-dimensional CNN baseline. Sparse 8-bit stochastic quantization with local error feedback reduces one upload to 61,051 bytes and the cumulative 50-round payload to 11.6 MB, against 952.5 MB for uncompressed federated averaging of the baseline. Gaussian update perturbation with Rényi accounting bounds the privacy loss of the shared updates at a weak client-level value ($\varepsilon \approx 24.8$), and the server combines the updates with a trimmed mean and Nesterov momentum, where trimming is inactive at four updates per round. Across five seeds, Fed-EdgeIDS achieved 98.74% ± 0.12% accuracy on Edge-IIoTset and 99.02% ± 0.09% on CIC-IDS2017, with false alarm rates of 0.62% and 0.49%. On a Raspberry Pi 4 and a Jetson Nano, per-window inference took 3.74 ms and 1.18 ms, and mean detection-to-dispatch latency was 4.56 ms and 1.82 ms.

Four directions are planned for future work. Evaluating Byzantine robustness with at least five updates per round would test the trimmed-mean aggregator. Asynchronous federated aggregation would let cellular gateways with intermittent links upload updates without synchronous round delays. Self-supervised learning on unlabeled plant traffic would reduce dependence on labeled attack captures. Hardware-aware neural architecture search would target ARM Cortex-M microcontrollers with less than 512 KB of SRAM.

AUTHOR CONTRIBUTION STATEMENT

Hegazi M. Ibrahim designed the hardware, performed the experiments, generated the results, and edited the manuscript. Ahmed Hammad formulated the theory and mathematical framework of the work, designed and implemented the computer simulations, and wrote the manuscript. Both authors read and approved the final version of the manuscript.

ETHICS APPROVAL AND CONSENT TO PARTICIPATE

This study used a publicly available, de-identified image dataset and involved no new data collection from human participants or animals. Therefore, ethical approval and consent to participate are not applicable.

CONSENT FOR PUBLICATION

Not Applicable.

DATA AVAILABILITY

The Edge-IIoTset and CIC-IDS2017 benchmark datasets analyzed during this study are publicly available in their respective institutional repositories. The preprocessing, Dirichlet partitioning, and federated simulation scripts are available from the corresponding author upon reasonable request.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the intellectual contribution of the broader interdisciplinary literature reviewed in this article. The authors also acknowledge the use of ChatGPT-5 for assistance with manuscript refinement.

FUNDING

No Funding.

DISCLOSURE STATEMENT

No competing interests are declared by the authors.

REFERENCES

[1] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.

[2] J. Lin, W. Yu, N. Zhang, X. Yang, H. Zhang, and W. Zhao, "A survey on internet of things: Architecture, enabling technologies, security and privacy, and applications," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1125–1142, 2017.

[3] N. Falliere, L. O Murchu, and E. Chien, "W32.Stuxnet Dossier," Security Response White Paper Version 1.4, Symantec Corporation, Feb. 2011.

[4] A. Cherepanov, "WIN32/INDUSTROYER: A New Threat for Industrial Control Systems," White Paper Version 2017-06-12, ESET, June 2017.

[5] Electricity Information Sharing and Analysis Center (E-ISAC), "Analysis of the Cyber Attack on the Ukrainian Power Grid," technical report, SANS Industrial Control Systems, Washington, DC, USA, Mar. 2016.

[6] A. Di Pinto, Y. Dragoni, and A. Carcano, "Triton: The first ics cyber attack on safety instrument systems: Understanding the malware, its communications and its ot payload," in *Proceedings of Black Hat USA*, (Las Vegas, NV, USA), pp. 1–26, 2018.

[7] Y. Liu, P. Ning, and M. K. Reiter, "False data injection attacks against state estimation in electric power grids," *ACM Transactions on Information and System Security*, vol. 14, no. 1, pp. 1–33, 2011.

[8] M. A. Ferrag, L. Maglaras, S. Moschoyiannis, and H. Janicke, "Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study," *Journal of Information Security and Applications*, vol. 50, p. 102419, Feb. 2020.

[9] International Electrotechnical Commission (IEC), "Communication Networks and Systems for Power Utility Automation — Part 5: Communication Requirements for Functions and Device Models." IEC 61850-5:2013, Edition 2.0, Jan. 2013. `https://webstore.iec.ch/en/publication/6012`.

[10] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[11] P. K. R. Maddikunta, Q.-V. Pham, P. B, N. Deepa, K. Dev, T. R. Gadekallu, R. Ruby, and M. Liyanage, "Industry 5.0: A survey on enabling technologies and potential applications," *Journal of Industrial Information Integration*, vol. 26, p. 100257, 2022.

[12] A. A. Diro and N. Chilamkurti, "Distributed attack detection scheme using deep learning approach for internet of things," *Future Generation Computer Systems*, vol. 82, pp. 761–768, 2018.

[13] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, "Edge intelligence: Paving the last mile of artificial intelligence with edge computing," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1738–1762, 2019.

[14] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of edge computing and deep learning: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 869–904, 2020.

[15] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y. Arcas, "Communication-Efficient Learning of Deep Networks from Decentralized Data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics* (A. Singh and J. Zhu, eds.), vol. 54 of *Proceedings of Machine Learning Research*, pp. 1273–1282, PMLR, 20–22 Apr 2017.

[16] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[17] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," in *Proceedings of the 4th International Conference on Learning Representations (ICLR)*, (San Juan, Puerto Rico), 2016.

[18] J. Konečný, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," in *Proceedings of the NIPS Workshop on Private Multi-Party Machine Learning*, 2016.

[19] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," in *Advances in Neural Information Processing Systems* (H. M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. B. Fox, and R. Garnett, eds.), vol. 32, (Red Hook, NY, USA), pp. 14774–14784, Curran Associates, Inc., 2019.

[20] P. Blanchard, E. M. El Mhamdi, R. Guerraoui, and J. Stainer, "Machine learning with adversaries: Byzantine tolerant gradient descent," in *Advances in Neural Information Processing Systems*, vol. 30, pp. 119–129, Curran Associates, Inc., 2017.

[21] M. Roesch, "Snort—lightweight intrusion detection for networks," in *Proceedings of the 13th USENIX Conference on System Administration (LISA '99)*, (Seattle, WA, USA), pp. 229–238, USENIX Association, 1999.

[22] V. Paxson, "Bro: A system for detecting network intruders in real-time," *Computer Networks*, vol. 31, no. 23–24, pp. 2435–2463, 1999.

[23] J. Tai, I. Alsmadi, Y. Zhang, and F. Qiao, "Machine learning methods for anomaly detection in industrial control systems," in *2020 IEEE International Conference on Big Data (Big Data)*, pp. 2333–2339, 2020.

[24] I. Ahmad, M. Basheri, M. J. Iqbal, and A. Rahim, "Performance comparison of support vector machine, random forest, and extreme learning machine for intrusion detection," *IEEE Access*, vol. 6, pp. 33789–33795, 2018.

[25] Y. AbdelSatar, F. E. Mohamed, S. AbdelNasser, and A. A. Mohamed, "Intelligent MITM Attack Detection Systems Using Ensemble Learning for IoT Network Security," *Engineering Systems and Intelligent Technologies (ESIT)*, vol. 3, no. 2, pp. 49–58, 2026.

[26] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[27] M. Z. Alom, V. Bontupalli, and T. M. Taha, "Intrusion detection using deep belief networks," in *2015 National Aerospace and Electronics Conference (NAECON)*, (Dayton, OH, USA), pp. 339–344, IEEE, June 2015.

[28] S. Kiranyaz, O. Avci, O. Abdeljaber, T. Ince, M. Gabbouj, and D. J. Inman, "1d convolutional neural networks and applications: A survey," *Mechanical Systems and Signal Processing*, vol. 151, p. 107398, 2021.

[29] 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.

[30] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[31] M. Al-Hawawreh, N. Moustafa, and E. Sitnikova, "Identification of malicious activities in industrial internet of things based on deep learning models," *Journal of Information Security and Applications*, vol. 41, pp. 1–11, Aug. 2018.

[32] 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.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention Is All You Need," in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, vol. 30, pp. 5998–6008, Curran Associates, Inc., 2017.

[34] Z. Wu, H. Zhang, P. Wang, and Z. Sun, "Rtids: A robust transformer-based approach for intrusion detection system," *IEEE Access*, vol. 10, pp. 64375–64387, 2022.

[35] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 12:1–12:19, 2019.

[36] V. Rey, P. M. Sánchez Sánchez, A. Huertas Celdrán, and G. Bovet, "Federated learning for malware detection in iot devices," *Computer Networks*, vol. 204, p. 108693, 2022.

[37] V. Mothukuri, P. Khare, R. M. Parizi, S. Pouriyeh, A. Dehghantanha, and G. Srivastava, "Federated-learning-based anomaly detection for iot security attacks," *IEEE Internet of Things Journal*, vol. 9, no. 4, pp. 2545–2554, 2022.

[38] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pp. 1–7, 2019.

[39] S. Wang, T. Tuor, T. Salonidis, K. K. Leung, C. Makaya, T. He, and K. Chan, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[40] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1800–1807, 2017.

[41] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," in *Low-Power Computer Vision: Improving the Efficiency of Artificial Intelligence* (P. Molchanov, T. Ajanthan, O. Mutlu, *et al.*, eds.), pp. 291–326, Chapman and Hall/CRC, 2022.

[42] A. Reisizadeh, A. Mokhtari, H. Hassani, A. Jadbabaie, and R. Pedarsani, "FedPAQ: A communication-efficient federated learning method with periodic averaging and quantization," in *Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics (AISTATS)*, vol. 108 of *Proceedings of Machine Learning Research*, pp. 2021–2031, PMLR, 2020.

[43] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified SGD with memory," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 31, pp. 4452–4463, Curran Associates, Inc., 2018.

[44] D. Yin, Y. Chen, R. Kannan, and P. Bartlett, "Byzantine-robust distributed learning: Towards optimal statistical rates," in *Proceedings of the 35th International Conference on Machine Learning* (J. Dy and A. Krause, eds.), vol. 80 of *Proceedings of Machine Learning Research*, pp. 5650–5659, PMLR, 10–15 Jul 2018.

[45] S. J. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," in *Proceedings of the 9th International Conference on Learning Representations (ICLR)*, 2021.

[46] M. Abadi, A. Chu, I. Goodfellow, H. B. McMahan, I. Mironov, K. Talwar, and L. Zhang, "Deep learning with differential privacy," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pp. 308–318, ACM, 2016.

[47] I. Mironov, "Rényi differential privacy," in *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*, pp. 263–275, 2017.

[48] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proceedings of Machine Learning and Systems 2 (MLSys 2020)* (I. Dhillon, D. Papailiopoulos, and V. Sze, eds.), vol. 2, pp. 429–450, 2020.

[49] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *2017 IEEE International Conference on Computer Vision (ICCV)*, pp. 2999–3007, IEEE, 2017.

[50] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, 2015.

[51] H. B. McMahan, D. Ramage, K. Talwar, and L. Zhang, "Learning differentially private recurrent language models," in *International Conference on Learning Representations (ICLR)*, 2018.

[52] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, vol. 30, pp. 1709–1720, Curran Associates, Inc., 2017.

[53] I. Sutskever, J. Martens, G. E. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," in *Proceedings of the 30th International Conference on Machine Learning*, vol. 28 of *Proceedings of Machine Learning Research*, pp. 1139–1147, PMLR, 2013.

[54] 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.

[55] I. Sharafaldin, A. Habibi Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, vol. 1, pp. 108–116, 2018.