Published by
NIGERIAN SOCIETY OF PHYSICAL SCIENCES
Available online @ https://asr.nsps.org.ng/index.php/asr

African Scientific Reports 5 (2026) 646

African Scientific Reports

Security-aware selective quantization and optimization framework (SASQOF) for resource-constrained ML-based IoT irrigation systems

Dauda Olorunkemi Isiaka*, Afolayan Ayodele Obiniyi, Victoria Ifeoluwa Yemi-Peters, Joshua Babatunde Agbogun

Department of Computer Science, Federal University Lokoja, Kogi State, Nigeria

Abstract

This study evaluates selective dynamic 8-bit integer (INT8) quantization of long short-term memory (LSTM) recurrent weights while retaining a 32-bit floating-point (FP32) linear output layer, together with authenticated sensor processing, using 10,257 observations from the Secured and Optimized IoT-Based Irrigation System (SOIBIS) experimental farm at Federal University Lokoja, Nigeria. Five seeded runs compare eight models under chronological and feature-group evaluations. The chronological test contains only label pattern 101, for which a majority baseline also achieves 100% exact-match accuracy. On the class-diverse feature-group test, selective quantization achieves 98.40% mean exact-match accuracy, compared with 100% for a decision tree and 84.68% for a majority baseline. Selective quantization reduces serialized model size by 64.9% and single-record host model-only latency by 50.7%, but increases 12-record model-only latency by approximately 41.1%. Mean authenticated single-record inference-path time is 0.441 ms on the host. All 1,200 specified functional packet checks met their expected outcomes. These are host-software findings, not evidence of ESP32 deployment, independently verified pump actuation, water savings, or agronomic superiority.

DOI: 10.46481/asr.2026.5.3.646

Keywords: Irrigation control, Selective quantization, Security-aware design, Sensor authentication, Edge machine learning

Article History:
Received: 28 August 2026
Received in revised form: 27 October 2026
Accepted for publication: 30 October 2026
Available online: 16 November 2026

© 2026 The Author(s). Published by the Nigerian Society of Physical Sciences under the terms of the Creative Commons Attribution 4.0 International license. Further distribution of this work must maintain attribution to the author(s) and the published article's title, journal citation, and DOI.

1. Introduction

Sensor-based irrigation control couples soil and environmental measurements with pump-operation decisions. Although machine learning (ML) can assist with these decisions, its usefulness for irrigation does not, by itself, establish classifier accuracy. Data collection, target construction, computational resources, and message integrity must be addressed individually. Oladipe *et al.* [1] identify real-time deployment and cross-regional validation among the remaining challenges for agricultural ML. Cloud dependence is possible but not inevitable, as demonstrated by the edge-first irrigation architecture of Taueatsoala *et al.* [2].

Alongside model storage, constrained inference requires support for computational operators and sufficient working memory. Lin et al. [3] address these requirements through joint architecture and inference-engine design. A small host-model file does not

*Corresponding Author Tel. No.: +234-703-449-9709.

Email address: dauda.isiaka@fulokoja.edu.ng (Dauda Olorunkemi Isiaka)

establish ESP32 compatibility, and host-processor timing cannot establish microcontroller inference timing. A hash-based message authentication code (HMAC) provides message integrity and shared-key authentication [4]. Replay resistance additionally requires receiver-maintained state and authenticated freshness information [5].

This study evaluates the Security-Aware Selective Quantization and Optimization Framework (SASQOF), an irrigation software framework developed in this research. It implements selective dynamic integer (INT8) quantization of long short-term memory (LSTM) weights while retaining a single-precision floating-point (FP32) output layer. Its contribution is application-level software integration and a reproducible evaluation of limitations, rather than a new quantizer, a hardware-aware precision-search method, or a cryptographic primitive.

The evaluation compares predictive accuracy against matched alternatives, along with serialized model size, host-processor latency, and authenticated-packet processing overhead. The dataset was audited before evaluation. Chronological testing is the primary analysis, while a separately identified sensitivity analysis examines class-diverse records and keeps identical feature vectors within one partition. Neither analysis constitutes a field-irrigation experiment.

2. Related work

2.1. Irrigation learning and constrained inference

Taueatsoala et al. [2] present an irrigation-control architecture using ESP32 offline inference and a Raspberry Pi local server. They use gradient boosting and assess performance with regression metrics. Akazue [6] presents fuzzy-logic irrigation control. These studies provide context but do not yield directly comparable accuracy or timing baselines.

TinyML research addresses computational-graph and memory compatibility [3]. Liu et al. [7] review lightweight deep learning in resource-constrained environments. These studies inform the assessment of field validation and sensor-data quality, but their reported performance is not used as a direct comparison with the irrigation classifiers evaluated here.

2.2. Quantization and compression

The effects of quantization on operator support, file size, and runtime depend on the method and execution backend and cannot be guaranteed [8–10]. Jacob et al. [8] discuss quantization-aware training and integer-only inference. Li and Alvarez [11] address recurrent-network quantization. Maniam et al. [12] propose an IoT-based RNN-LSTM model for soil-moisture prediction but do not establish INT8 recurrent-weight compression or a quantized microcontroller implementation. These studies address different tasks and are not used for direct performance ranking.

Pruning removes network connections or structures, whereas quantization changes numerical representation. Han et al. [13] describe multistage hybrid compression using pruning, trained quantization, and coding. The present work does not implement pruning, quantization-aware training, or automated precision search. Instead, selective conversion uses the documented PyTorch dynamic-quantization interface [9] and dynamically quantized LSTM module [10], which retain floating-point external tensor interfaces. Host CPU testing confirms PyTorch conversion on the tested host, not ESP32 operator compatibility.

2.3. Sensor authentication

Under the assumed secrecy of the shared key, HMAC provides message integrity and shared-key authentication [4]. It does not encrypt messages or establish the physical accuracy of data from a compromised legitimate sensor. Replay prevention also requires freshness management and receiver state [5]. This study specifies a packet protocol, integrates verification with inference, and measures host-processor overhead. Functional rejection tests establish reference-software behavior under the tested conditions, not resistance to physical key extraction or endpoint compromise.

3. Materials and methods

3.1. Architecture and implementation scope

The implemented host-processing path is shown in Figure 1. Each packet contains the sensor payload, originating device identity, key epoch, sequence number, and timestamp. The receiver checks message integrity and freshness before forwarding measurements to preprocessing and classification. The three classifier outputs represent pump-activation labels. No physical actuator was operated in these experiments. Remaining controller-integration tasks include agronomic safeguards, interlocks, and safe responses to rejected or missing readings.

The reported tests were performed on a Linux x86_64 host with an AMD EPYC 9V74 80-Core Processor, using one CPU thread, Python 3.12.14, PyTorch 2.14.1+cpu, NumPy 2.3.5, pandas 2.2.3, and scikit-learn 1.8.0. Evaluation results and archived outputs are summarized in Tables 3, 6, and 8. Model-only and preprocessing-inclusive timings are distinct measurements and must not be interchanged. No ESP32 inference timing, flash or RAM footprint, or physical actuation measurements are claimed.

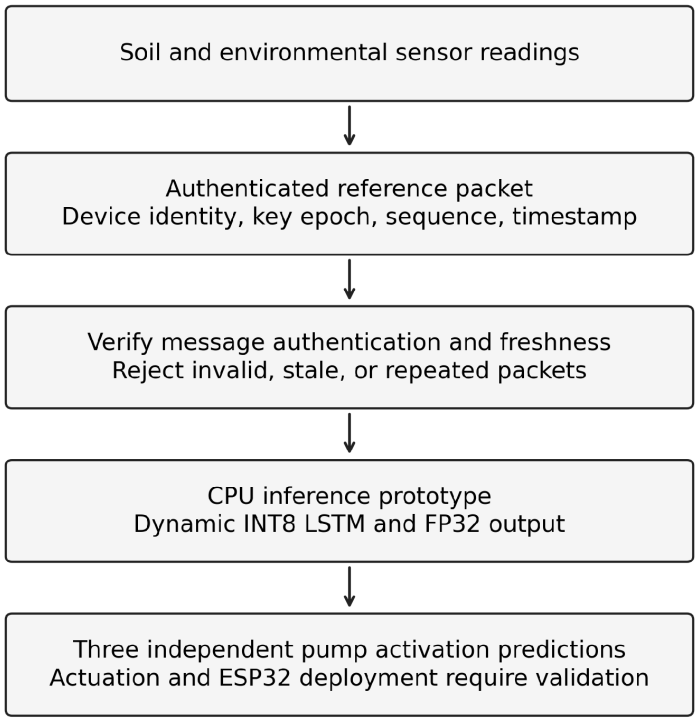


Figure 1: Authenticated host reference architecture. Physical actuation and ESP32 deployment were not evaluated.

3.2. Dataset audit and input definitions

The study used an experimental dataset comprising 10,257 observations and 13 columns collected at the Federal University Lokoja experimental farm in Felele, Kogi State, Nigeria, using the Secured and Optimized IoT-Based Irrigation System (SOIBIS) setup. The coordinates 7°50′05″–7°54′04″ N and 6°40′22″–6°42′02″ E describe the surrounding area rather than measured sensor-installation coordinates. The acquisition platform used an ESP32-S3 DevKitC-1: GPIO pins 1, 2, and 3 monitored the sandy, clay, and loamy soil-moisture channels, respectively; a DHT11 monitored ambient temperature and relative humidity; and a BMP180 measured barometric pressure and temperature over I2C. An MPU6050 accelerometer/gyroscope supported movement and tamper monitoring. Three active-HIGH relay outputs (GPIO pins 5, 6, and 7) drove the corresponding irrigation-pump channels. An SPI SD-card module, serial monitor, and web-based dashboard supported inspection and notification.

The supplied hardware inventory did not identify individual capacitive-probe manufacturers or part numbers. Raw analog-to-digital converter (ADC) output spans dimensionless counts from 0 to 4,095 (12 bits). Firmware maps each raw count to a relative moisture percentage using moisture (%) = $100 \times (\text{rawDry} - \text{rawReading})/(\text{rawDry} - \text{rawWet})$, clamped to 0–100%. The sketch specifies default endpoints rawDry = 3,000 and rawWet = 1,500, with channel-specific calibration supported. Although calibration records were reported, these values are software defaults rather than verified measured calibration points. The variables sandy_percent, clay_percent, and loamy_percent refer to relative soil-zone moisture indices, not proportions of soil types or independently validated volumetric soil-water content.

The source records do not establish physical units for the raw ADC-related values; temperature and humidity readings are recorded in their respective reported scales. The binary sandy, clay, and loamy controller targets exactly match the recorded binary decisions in all 10,257 observations. A moisture-below-50% rule is consistent with all three recorded targets, but does not independently establish the historical controller configuration or physical pump state. The dataset contains nominal two-minute samples over approximately 15 days. Consequently, the model scores quantify reproduction of recorded controller decisions rather than measured water savings, agronomic optimality, or physical pump operation.

All seven numerical predictors and their audited distributions are shown in Table 1. The three soil-zone channels are measured simultaneously rather than representing mutually exclusive categories; each appears in all 10,257 records. Their percentage-based names do not, by themselves, establish soil composition or volumetric water content. The recorded numerical values are used as supplied, without assigning unsupported physical units to environmental measurements. The binary rain indicator is positive in only

Table 1: Input schema and recorded distribution across all 10,257 rows.

Input	Type	Recorded range	Mean ± SD	Preprocessing
sandy_percent	Numeric	0.00 to 98.00	5.25 ± 13.48	Training only scaling
clay_percent	Numeric	0.00 to 77.00	59.79 ± 7.05	Training only scaling
loamy_percent	Numeric	0.00 to 63.53	15.24 ± 15.64	Training only scaling
temperature	Numeric	0.00 to 36.50	31.11 ± 3.07	Training only scaling
humidity	Numeric	0.00 to 100.00	75.10 ± 13.75	Training only scaling
dew_point	Numeric	0.00 to 26.86	25.85 ± 0.86	Training only scaling
is_raining	Binary 0/1	0.00 to 1.00	0.00 ± 0.02	Retain binary

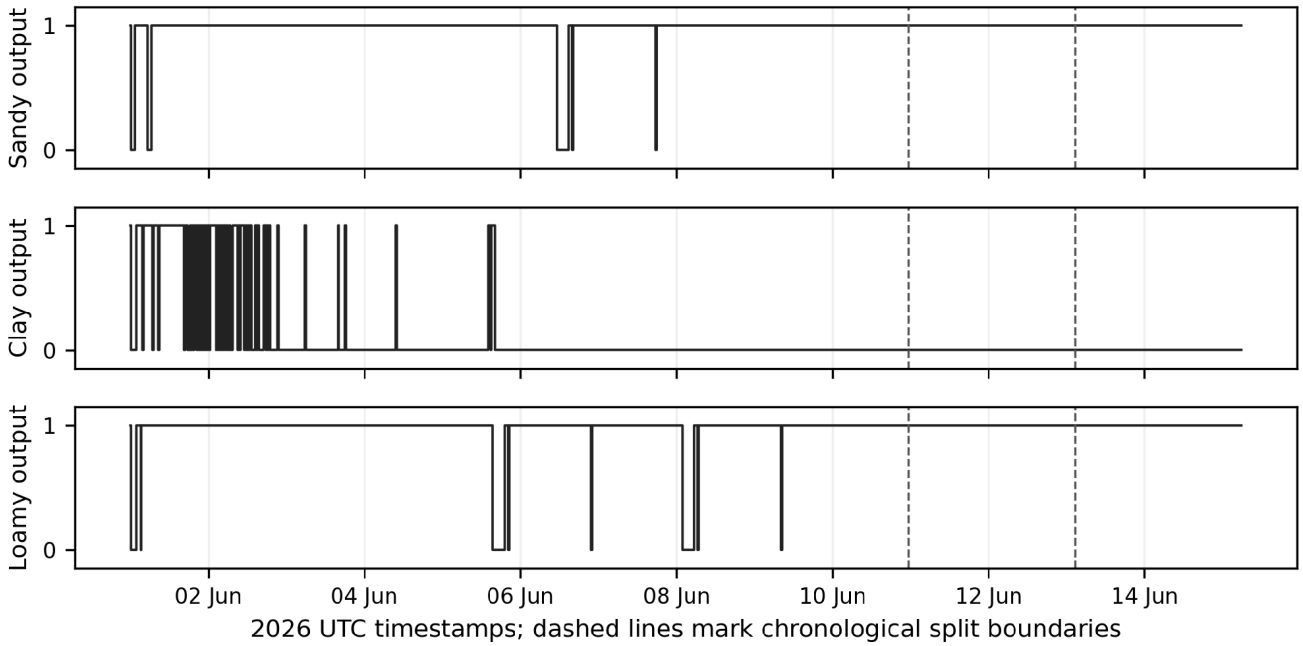


Figure 2: Observed binary label timeline. Both chronological validation and test periods contain only pattern 101.

five records, all in the chronological training partition.

No missing or non-finite predictor values, duplicate timestamps, or duplicate identifiers were detected. No contradictory labels were found for identical feature vectors.

Timestamps range from 1 June 2026 at 00:00:00 UTC to 15 June 2026 at 05:51:08.720 UTC. Intervals range from 119.795 to 120.195 seconds, with a median of 119.995 seconds. Dates were not inferred from the source filename. Identifier and datetime fields were excluded as predictors. For every record, the normalized IRRIGATION PATTERN field equals 100 times the sandy label plus 10 times the clay label plus the loamy label; it was therefore excluded to avoid target leakage. Trailing spaces were removed from source column names.

Positive label counts are 10,056 for sandy, 948 for clay, and 9,938 for loamy. Patterns 101, 111, 100, 001, 000, 011, and 110 occur 8,890, 882, 253, 131, 35, 35, and 31 times, respectively. All 10,257 records are consistent with a soil-moisture-below-50% rule for the three binary labels. This relationship supports threshold-consistent controller decisions, not independently observed relay or pump activity. The source-code defaults do not independently establish the precise thresholds active during data collection. Figure 2 shows the chronology of the recorded binary labels.

The last non-101 record was identifier 6,015, dated 9 June 2026 at 08:27:29.930 UTC. The remaining 4,242 records have pattern 101. Figure 2 illustrates the lack of label diversity in the chronological validation and test periods. Additional data covering different operating regimes are needed to evaluate minority decisions and generalization.

3.3. Prediction task and model training

The prediction task is multilabel classification of three parallel binary pump-activation decisions. Pattern 111 occurs in 882 records. The three targets retain their original binary values: 0 (OFF decision) and 1 (ON decision). A sigmoid transformation

Table 2: Implemented precision and authentication ablations.

Configuration	Converted modules	Interface and output scope
FP32 LSTM	None	FP32 recurrent and linear weights; floating point outputs
Broad dynamic INT8	nn.LSTM and nn.Linear	Quantized packed weights; floating point input and outputs
Selective dynamic INT8	nn.LSTM only	Quantized recurrent weights; FP32 linear weights and sigmoid
Authenticated selective path	Same selective model	Identical weights; verify packet and freshness before preprocessing

converts each logit into a probability, and the fixed decision threshold is 0.5. These labels represent recorded controller decisions, not independently observed pump operation. The precision configurations are summarized in Table 2.

The LSTM contains seven input features, two recurrent layers with 32 hidden units each, and a linear mapping from 32 hidden features to three outputs. The hidden state is reinitialized independently for each window. The primary model uses one record per example and is therefore an explicitly single-step recurrent comparator, not a temporal-learning model. The 12-record sensitivity analysis covers approximately 22 minutes from the first to the last reading. It tests sensitivity to ordered history and computational cost but does not establish an agronomically optimal horizon.

Models were trained for 10 epochs with Adam optimization, seeds 42–46, a learning rate of 0.01, and a batch size of 64. Binary cross-entropy with logits was minimized; an equivalent probability formulation is given in Eq. (3). The checkpoint with the lowest validation loss was retained. The constant label pattern in the primary validation period limits this selection criterion, a limitation acknowledged without using test results to modify the checkpoint selection. Sequence lengths and the decision threshold were protocol choices rather than test-set optimizations.

The eight comparators were a per-output training-majority baseline, a depth-5 decision tree, a random forest with 10 trees and maximum depth 10, logistic regression, a multilayer perceptron (MLP) with two hidden layers of width 32, an FP32 LSTM, a dynamically quantized INT8 LSTM with a quantized linear layer, and a selective dynamically quantized INT8 LSTM. For nonrecurrent baselines, window inputs were flattened. Neural architectures were not parameter matched. Comparisons used identical partition memberships and training-only preprocessing.

3.4. Precision configurations

Affine quantization uses a positive scale, s > 0, and an integer zero point, z, to represent a floating-point weight, W, as expressed in Eq. (1) [8]. The quantized integer q is limited by the representable range and backend rounding and clamping. The implemented PyTorch dynamic scheme quantizes recurrent LSTM weights while retaining FP32 output-layer weights and a floating-point external interface [9, 10].

$$W_q = \mathrm{clip}\left(\mathrm{round}(W/s) + z, q_{\min}, q_{\max}\right), \qquad \widehat{W} = s(W_q - z). \tag{1}$$

Selective dynamic quantization targets only nn.LSTM [9, 10]. The linear output weights, bias, and sigmoid remain FP32. Input tensors and returned hidden features use floating-point representations; internal recurrent states, biases, accumulators, and nonlinear operations are not all characterized as INT8. In Eq. (2), X denotes an input window, h_T its final returned hidden feature, and p the three output probabilities. The equation preserves the implemented FP32 output path.

$$h_T = \mathrm{LSTM}_{\mathrm{dyn\,INT8}}(X)_T, \qquad p = \sigma\left(W_{\mathrm{out,FP32}} h_T + b_{\mathrm{out,FP32}}\right). \tag{2}$$

For binary label y_{ij} and predicted probability p_{ij}, the mean loss over N examples and three outputs is:

$$L = -\frac{1}{3N} \sum_{i=1}^{N} \sum_{j=1}^{3} \left[y_{ij} \log p_{ij} + (1 - y_{ij}) \log(1 - p_{ij}) \right]. \tag{3}$$

Table 2 lists the implemented ablations. For each seed, the same selected FP32 checkpoint was cloned and converted into both quantized configurations. The supplementary material contains the training, checkpoint selection, conversion, prediction, and export procedures. The evaluated implementation uses dynamically quantized host recurrent weights, floating-point interfaces, and an FP32 output layer in the selective configuration.

3.5. Packet authentication and security checks

The attacker is assumed able to observe, modify, inject, delay, and replay packets but unable to obtain secret keys or modify the trusted receiver. Under these assumptions, HMAC provides shared-key integrity and authentication [4]. It does not provide confidentiality, verify physical sensor measurements, protect endpoints, or prevent all denial-of-service attacks. The inner and outer pads, ipad and opad, appear in Eq. (4); the symbol ⊕ denotes exclusive OR and || denotes concatenation.

$$\mathrm{HMAC}_K(m) = H\left((K' \oplus \mathrm{opad}) \,\|\, H\left((K' \oplus \mathrm{ipad}) \,\|\, m\right)\right). \tag{4}$$

The reference receiver authenticates the packet bytes covering device identity, authorized key epoch, strictly increasing sequence number, timestamp, and payload. It uses constant-time tag comparison, a maximum packet age of 60 seconds, and a permitted future offset of 10 seconds. It atomically checks and commits a persistent per-device, per-epoch sequence high-water mark. Failed authentication does not advance this state. The stored sequence state remains available after receiver restart; an incoming packet cannot authorize a new key epoch.

A shared key is supplied externally to the receiver. Each benchmark uses a cryptographically random 32-byte key file; no default production key is embedded. Deployment requirements include secure provisioning, protected key storage, authenticated key rotation and revocation, persistent sender state, and trusted clocks. Physical provisioning and protection were not evaluated. A delayed first delivery may be accepted if it still satisfies the age window.

One hundred payloads were tested against each of 12 functional cases: valid acceptance, payload modification, wrong key, duplicate replay, replay after restart, expired timestamp, future timestamp, revoked epoch, unknown device, malformed schema, failed-tag state preservation, and out-of-order sequence. Valid packets and the valid follow-up packets in the state-preservation case were expected to be accepted; the other packets were expected to be rejected. These tests establish protocol behavior under stated conditions, not field-network attack-success probabilities.

3.6. Partitioning metrics and measurement protocol

Figure 3 summarizes the executed host evaluation workflow.

The primary chronological split assigned 7,179, 1,538, and 1,540 observations to training, validation, and test partitions, respectively. Because the split is time-based, identical predictor vectors can occur in different partitions.

For sequence length T = 1, the partitions yield 7,179, 1,538, and 1,540 single-record examples. For T = 12, windows were constructed separately within each partition, excluding the first 11 possible windows per partition and yielding 7,168, 1,527, and 1,529 windows. No window crossed a partition boundary. Sixty samples were inspected to verify approximate two-minute sampling continuity. Example counts for T = 1 and T = 12 are therefore not interchangeable.

In a separate record-level sensitivity analysis, observations with identical values across all seven raw predictor features were grouped before partitioning. The 1,156 unique feature groups were stratified by observed target pattern and allocated using fixed split seed 42. Training, validation, and test partitions contained 805, 172, and 179 feature groups, corresponding to 7,244, 1,564, and 1,449 observations. All observations from a given group remained in one partition. An audit of saved memberships confirmed that no identical raw seven-feature vectors crossed partition boundaries. The method does not remove near-duplicates or temporal dependence and is therefore a sensitivity analysis rather than a chronological assessment of future performance.

A training-only median imputer was fitted as the specified missing-value policy, although no missing values required imputation. Six continuous predictors were scaled using training-set means and variances; the rain indicator remained binary. Binary targets, identifiers, timestamps, and the combined label code were not scaled or used as predictors. Prediction decisions follow Eq. (5), and exact-match and per-label accuracy are defined separately in Eq. (6).

$$\widehat{y_{ij}} = \mathbf{1}[p_{ij} \geq \tau_j], \qquad \tau_j = 0.5, \tag{5}$$

$$A_{\mathrm{subset}} = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[\widehat{y_i} = y_i], \qquad A_{\mathrm{label}} = \frac{1}{3N}\sum_{i=1}^{N}\sum_{j=1}^{3} \mathbf{1}[\widehat{y_{ij}} = y_{ij}]. \tag{6}$$

Precision, recall, F1, specificity, false-positive rate, and false-negative rate were calculated for each output from true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), as shown in Eqs. (7) and (8). Macro F1 is the unweighted mean of the three per-output F1 values. Measures with zero denominators were reported as undefined and exported as JSON null; macro values containing an undefined component were likewise reported as undefined rather than assigning unsupported positive-class performance.

$$\begin{aligned} P_j &= \frac{TP_j}{TP_j + FP_j}, \qquad R_j = \frac{TP_j}{TP_j + FN_j}, \\ F_{1,j} &= \frac{2TP_j}{2TP_j + FP_j + FN_j}, \qquad F_{1,\mathrm{macro}} = \frac{1}{3}\sum_{j=1}^{3} F_{1,j}. \end{aligned} \tag{7}$$

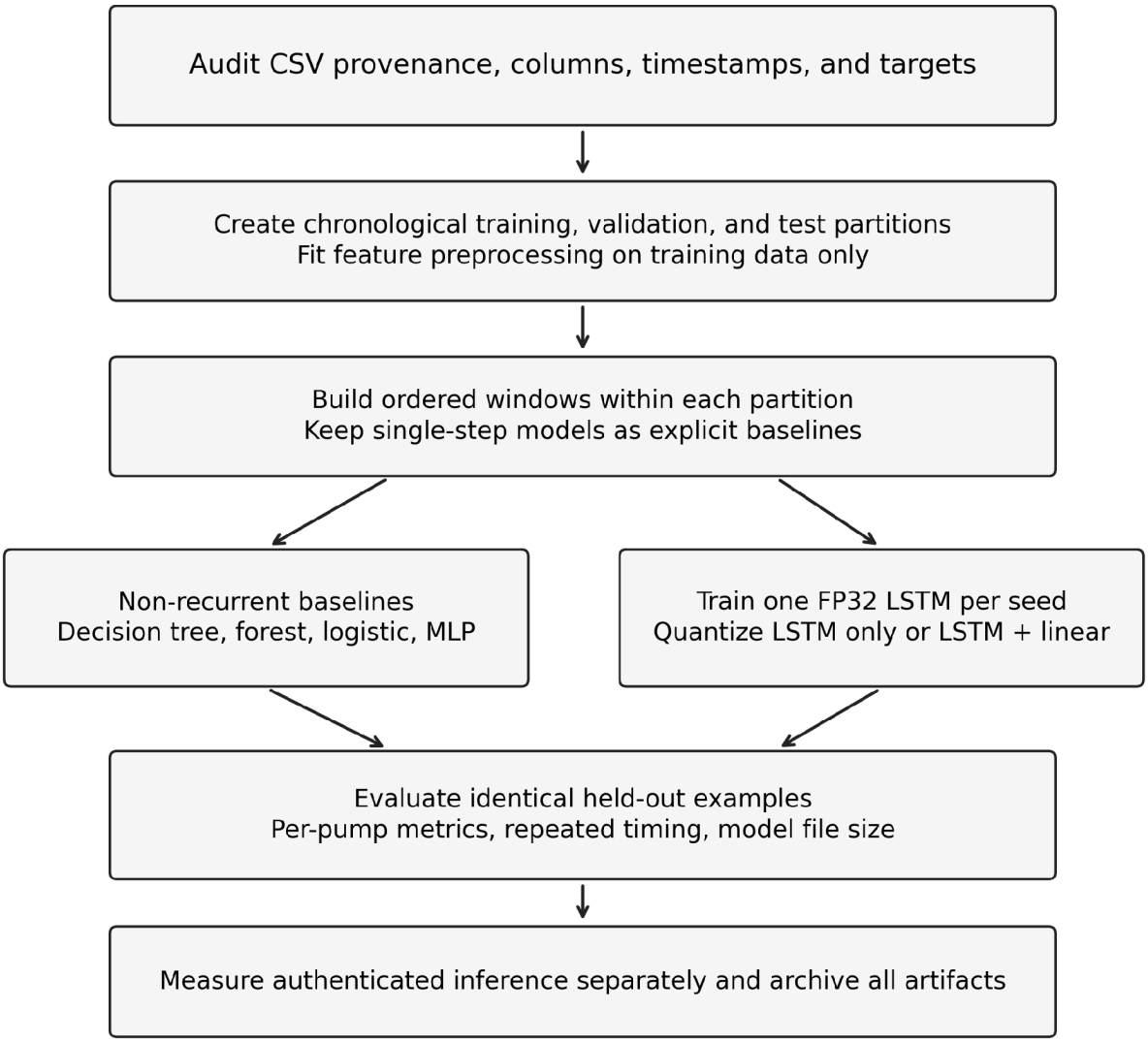


Figure 3: Executed host evaluation workflow. Chronological testing is primary; a separate input group sensitivity is restricted to single records.

$$S_j = \frac{TN_j}{TN_j + FP_j}, \qquad \mathrm{FPR}_j = \frac{FP_j}{FP_j + TN_j}, \qquad \mathrm{FNR}_j = \frac{FN_j}{FN_j + TP_j}. \tag{8}$$

Prediction results for fixed partitions are summarized as means and sample standard deviations across five training seeds (42–46). Exploratory paired-seed intervals and 2,000 feature-group bootstrap resamples for seed 42 describe variation within the specified evaluation. Because feature groups may remain dependent through time or environmental conditions, these intervals are conditional and should not be interpreted as unconditional confidence bounds for deployment.

Model-only timing includes the model forward pass and neural sigmoid but excludes preprocessing, packet handling, network transmission, and actuation. Measurements used batch size one, evaluation mode, no gradients, one CPU thread, 50 warm-up calls, and 500 timed calls per model and seed, measured with perf_counter_ns. Means, within-run sample standard deviations, medians, and 95th percentiles were archived. Neural state dictionaries were serialized with torch.save, while classical models were serialized with joblib. Different serializers limit direct file-size comparisons; serialized bytes do not measure peak working RAM, firmware flash, or SRAM.

The matched selective inference paths include feature construction, preprocessing, prediction, and thresholding, with or without packet verification, clock checks, and durable sequence commits. Signing, network delay, and actuation were excluded from timing. Authenticated packets contained 240–244 payload bytes plus a 32-byte tag. Window maintenance and history acquisition were not timed; longer windows used trusted cached history, and only the current record was authenticated. Accordingly, the observed

Table 3: Chronological single record results and host resource measurements across five runs.

Model	Exact match % mean ± SD	Model time ms mean ± SD	File KiB mean ± SD
Majority	100.00 ± 0.00	0.008 ± 0.000	0.84 ± 0.00
Decision tree	100.00 ± 0.00	0.035 ± 0.002	2.87 ± 0.00
Random forest	100.00 ± 0.00	0.549 ± 0.151	61.96 ± 6.66
Logistic	100.00 ± 0.00	0.093 ± 0.002	1.68 ± 0.00
MLP FP32	100.00 ± 0.00	0.021 ± 0.001	8.53 ± 0.00
LSTM FP32	100.00 ± 0.00	0.152 ± 0.022	57.85 ± 0.00
LSTM INT8 recurrent + linear	100.00 ± 0.00	0.085 ± 0.002	20.56 ± 0.00
Selective INT8 LSTM	100.00 ± 0.00	0.075 ± 0.001	20.33 ± 0.00

Table 4: Input group sensitivity prediction metrics across five seeded runs.

Model	Exact match % mean ± SD	Per label % mean ± SD	Macro F1 mean ± SD
Majority	84.68 ± 0.00	94.41 ± 0.00	0.6511 ± 0.0000
Decision tree	100.00 ± 0.00	100.00 ± 0.00	1.0000 ± 0.0000
Random forest	99.65 ± 0.50	99.88 ± 0.17	0.9994 ± 0.0009
Logistic	98.90 ± 0.00	99.63 ± 0.00	0.9874 ± 0.0000
MLP FP32	96.67 ± 0.80	98.89 ± 0.27	0.9552 ± 0.0203
LSTM FP32	97.28 ± 1.05	99.09 ± 0.35	0.9668 ± 0.0252
LSTM INT8 recurrent + linear	98.40 ± 0.68	99.47 ± 0.23	0.9917 ± 0.0133
Selective INT8 LSTM	98.40 ± 0.68	99.47 ± 0.23	0.9917 ± 0.0133

difference is not a universal estimate of security overhead.

4. Results

4.1. Chronological evaluation and dataset limitations

The dataset limitations and chronological evaluation are described below:

Only label pattern 101 occurs in the chronological validation and test partitions. Consequently, all tested models, including the majority baseline, achieved 100% exact-match accuracy on this limited interval. For sandy and loamy outputs, all 1,540 test labels are positive (TP = 1,540), so specificity and false-positive rate are undefined because there are no negative cases. For clay, TN = 1,540 and TP = FP = FN = 0; specificity is 1.000 and the false-positive rate is 0.000, while positive-class recall is undefined. These results do not assess performance across diverse label patterns.

Table 3 summarizes host resource measurements. Selective INT8 reduced mean single-record model-only inference time from 0.152 ms to 0.075 ms, a 50.7% reduction. Serialized model size fell from 59,237 bytes to 20,821 bytes, a 64.9% reduction. Broad dynamic INT8 produced a 21,051-byte file, slightly larger because of additional packed linear-layer metadata. File size also reflects serialization and cannot be inferred from nominal weight precision alone. The MLP was faster (0.021 ms) and smaller (8,737 bytes) than the recurrent comparators.

4.2. Class diverse sensitivity and per output errors

Table 4 reports the feature-group sensitivity results for all eight models on 1,449 test records. Seven label patterns occurred, with 1,397 positive sandy labels, 111 positive clay labels, and 1,369 positive loamy labels. Selective INT8 achieved 98.40% ± 0.68% mean exact-match accuracy compared with 97.28% ± 1.05% for FP32. The two dynamically quantized configurations produced identical predictions for each seed, so retaining an FP32 output layer showed no additional predictive benefit over broad dynamic quantization on these records. The decision tree achieved 100% accuracy in all five runs, consistent with the observed threshold-based label structure.

The exploratory paired mean difference between selective INT8 and FP32 was 1.12 percentage points, with a 95% seed-variation interval of 0.34 to 1.90 points. This interval conditions on one split and captures training randomness, not unresolved sensor dependence or dataset provenance. Seed 42 selective exact-match accuracy was 98.83%, with a conditional feature-group bootstrap interval of approximately 96.99% to 100%. Identical predictions from the two quantized configurations yielded a zero paired difference on these records but do not demonstrate general equivalence. Table 5 provides the corresponding representative per-pump counts.

Table 5: Per pump confusion counts and metrics on the input group test for seed 42.

Model	Pump	TP/TN/FP/FN	Precision	Recall	F1	Specificity
Majority	Sandy	1397/0/52/0	0.964	1.000	0.982	0.000
Majority	Clay	0/1338/0/111	Undef.	0.000	0.000	1.000
Majority	Loamy	1369/0/80/0	0.945	1.000	0.972	0.000
Tree	Sandy	1397/52/0/0	1.000	1.000	1.000	1.000
Tree	Clay	111/1338/0/0	1.000	1.000	1.000	1.000
Tree	Loamy	1369/80/0/0	1.000	1.000	1.000	1.000
Forest	Sandy	1397/52/0/0	1.000	1.000	1.000	1.000
Forest	Clay	111/1338/0/0	1.000	1.000	1.000	1.000
Forest	Loamy	1369/71/9/0	0.993	1.000	0.997	0.887
Logistic	Sandy	1397/52/0/0	1.000	1.000	1.000	1.000
Logistic	Clay	111/1330/8/0	0.933	1.000	0.965	0.994
Logistic	Loamy	1369/72/8/0	0.994	1.000	0.997	0.900

Table 5: Per pump confusion counts and metrics on the input group test for seed 42 (continued).

Model	Pump	TP/TN/FP/FN	Precision	Recall	F1	Specificity
MLP	Sandy	1396/52/0/1	1.000	0.999	1.000	1.000
MLP	Clay	111/1318/20/0	0.847	1.000	0.917	0.985
MLP	Loamy	1369/63/17/0	0.988	1.000	0.994	0.787
FP32 LSTM	Sandy	1394/52/0/3	1.000	0.998	0.999	1.000
FP32 LSTM	Clay	111/1319/19/0	0.854	1.000	0.921	0.986
FP32 LSTM	Loamy	1369/63/17/0	0.988	1.000	0.994	0.787
Broad INT8	Sandy	1397/52/0/0	1.000	1.000	1.000	1.000
Broad INT8	Clay	111/1338/0/0	1.000	1.000	1.000	1.000
Broad INT8	Loamy	1369/63/17/0	0.988	1.000	0.994	0.787
Selective INT8	Sandy	1397/52/0/0	1.000	1.000	1.000	1.000
Selective INT8	Clay	111/1338/0/0	1.000	1.000	1.000	1.000
Selective INT8	Loamy	1369/63/17/0	0.988	1.000	0.994	0.787

Table 5 presents integer confusion counts for the representative seed 42 rather than counts averaged across seeds.

Among 80 actual negative loamy-soil cases, the selective model correctly classified 63 as negative and incorrectly classified 17 as positive. Thus, specificity was 78.75%, and the false-positive rate was 21.25%.

$$\text{Specificity} = \frac{TN}{TN + FP} = \frac{63}{63 + 17} = 0.7875,$$

$$\text{False-positive rate} = \frac{FP}{FP + TN} = \frac{17}{17 + 63} = 0.2125.$$

The selective model incorrectly predicted pump activation for 21.25% of the actual negative loamy-soil cases and correctly identified 78.75%. This weakness in identifying negative cases under the seed-42 evaluation condition demonstrates why class-specific metrics, including specificity and false-positive rate, should be considered alongside overall accuracy.

4.3. Window sensitivity and runtime ablation

All 1,529 chronological 12-record test windows had the same label pattern, yielding 100% exact-match accuracy across models without establishing temporal learning. Table 6 compares host neural-model timing at two window lengths. Selective recurrent quantization was faster for single-record inputs but slower for 12-record windows on this backend; the difference is consistent with different execution costs for FP32 kernels and dynamically quantized operators, although kernel-level causes were not measured.

For 12-record windows, selective INT8 model-only inference averaged 0.254 ms compared with 0.180 ms for FP32, an increase of $(0.254 - 0.180)/0.180 \times 100 \approx 41.1\%$. This latency increase differs from the 64.9% reduction in serialized model size. For

Table 6: Host neural model time for single and twelve record windows across five runs.

Model	T = 1 ms mean ± SD	T = 12 ms mean ± SD
MLP FP32	0.021 ± 0.001	0.022 ± 0.001
LSTM FP32	0.152 ± 0.022	0.180 ± 0.016
LSTM INT8 recurrent + linear	0.085 ± 0.002	0.266 ± 0.024
Selective INT8 LSTM	0.075 ± 0.001	0.254 ± 0.021

Table 7: Functional checks using sensor payloads from the CSV.

Case	Expected behavior	Trials	Checks passed
Valid packets	Accept	100	100
Payload modification	Reject	100	100
Wrong signing key	Reject	100	100
Duplicate replay	Reject	100	100
Replay after restart	Reject	100	100
Expired timestamp	Reject	100	100
Future timestamp	Reject	100	100
Revoked key epoch	Reject	100	100
Unknown device	Reject	100	100
Malformed schema	Reject	100	100
Failed tag preserves state	Accept valid next packet	100	100
Out of order sequence	Reject	100	100

Table 8: Selective inference path timing on the host processor across five runs.

Window	Measured path	Mean ms ± SD	Mean of medians ms	Mean of P95 ms
1	Preprocess and infer	0.313 ± 0.019	0.292	0.408
1	Verify, preprocess and infer	0.441 ± 0.006	0.424	0.543
12	Preprocess and infer	0.508 ± 0.012	0.475	0.669
12	Verify, preprocess and infer	0.639 ± 0.022	0.602	0.826

single-record inference, selective INT8 was faster (0.075 versus 0.152 ms), demonstrating dependence on sequence length and host backend. Network transmission and physical actuation were not measured.

4.4. Packet checks and authenticated inference

All 100 functional checks for each of the 12 cases in Table 7 met their expected outcomes. Twelve additional unit tests also passed. These results establish the specified behavior of the reference implementation under the tested keys, times, and receiver-state assumptions. They do not estimate cryptographic forgery probability, detect physical sensor tampering, or demonstrate field adversarial resilience. Table 8 reports the associated inference-path timings.

Table 8 distinguishes model-only timing from preprocessing-inclusive and authenticated inference-path timing. Single-record authenticated inference averaged 0.441 ± 0.006 ms, compared with 0.313 ± 0.019 ms without packet verification. The approximate 0.128-ms difference includes parsing, tag validation, freshness checks, and durable sequence handling. For 12-record windows, authenticated and unauthenticated path means were 0.639 ± 0.022 ms and 0.508 ± 0.012 ms, respectively. These host-local measurements used cached trusted history and excluded ESP32 execution, network round trips, and actuator response.

4.5. Contextual literature comparison

Table 9 summarizes the relevant literature without ranking results across incompatible study designs.

5. Discussion and limitations

The storage and runtime evaluation showed that selective dynamic quantization reduced recurrent-model file size but did not reduce 12-record inference latency on the tested host. Both quantized configurations yielded identical predictions on the evaluated

Table 9: Literature context without a cross study performance ranking.

Study	Task and method	Implementation scope	Comparison boundary
Taueatsoala et. al. [2]	Gradient boosting regression; reported R^2 0.9973 and MAPE 0.99%	Edge first ESP32 and Raspberry Pi	Different task and protocol
Akazue [6]	Fuzzy logic irrigation decisions	Original 2016 study	No invented accuracy or latency
Maniam et. al. [10]	IoT-based RNN-LSTM soil-moisture prediction; no reported INT8 compression	IoT-based soil-moisture prediction system; MCU quantization not established	Different tasks and specifically for one protocol. No comparison with other peers
This study	Three label classification and packet verification	Measured host CPU reference	Controlled local comparisons only

records, while the observed input–label threshold relationship was reproduced most accurately by the decision tree. Retaining the FP32 output layer did not improve classification on these data, and the MLP was faster than the LSTM variants. Simpler models therefore provide relevant comparators for subsequent controller development.

The observed label-to-input relationship, scarcity of rain events, and constant chronological test pattern limit conclusions about agronomic prediction. Feature-group partitioning prevents exact seven-feature-vector overlap between partitions but does not eliminate temporal dependence, near duplicates, or target-construction bias. Longer windows and repeated seeds assess sensitivity within the supplied data; they do not establish independent forecasting or generalization to new farms or seasons.

The targets are threshold-consistent binary controller decisions, not independently verified records of pump operation. All 10,257 labels agree with the corresponding soil-moisture-below-50% relationship. Neither the historical label-generation configuration nor physical actuation can be inferred solely from this agreement.

Authentication and predictive validity are distinct: a genuine sensor may supply erroneous measurements, and an authenticated packet is not necessarily safe to use for actuation. The reference implementation demonstrated replay rejection and state persistence under trusted-key and clock assumptions. Deployment must still evaluate key provisioning, secure boot, hardware key protection, sender persistence, rollback resistance, resource exhaustion, and real wireless conditions. Elliptic-curve Diffie–Hellman is a key-agreement protocol rather than a digital-signature method; its use would require authenticated peers and a complete protocol design.

Host latency and serialized model bytes do not establish ESP32 deployability, peak working memory, power consumption, or firmware size. Future work requires independent data from varied operating conditions, matched physical controller trials, and an embedded implementation with explicit memory and timing measurements.

6. Conclusion

This study evaluates selective dynamic INT8 LSTM-weight quantization with an FP32 output layer and authenticated sensor processing on 10,257 SOIBIS observations. On the AMD EPYC 9V74 host, selective quantization reduced serialized model size by 64.9% and single-record model-only latency by 50.7%, but increased 12-record model-only latency by approximately 41.1% (0.254 versus 0.180 ms). The two dynamically quantized configurations made identical predictions on the evaluated records, without demonstrating universal equivalence. The chronological test was restricted to one label pattern and therefore did not distinguish the majority baseline from learned models. In the class-diverse sensitivity test, the majority baseline achieved 84.68% exact-match accuracy, the decision tree 100%, and selective INT8 98.40% on average. All 1,200 specified functional packet checks passed. These findings support a reproducible host-software comparison, not superior agronomic decisions, verified pump operation, or validated ESP32 deployment. Independent field data, documented calibration, physical controller trials, and embedded-resource measurements remain necessary.

Data availability

The analyzed CSV was supplied publicly by the corresponding author for this revision. Its SHA-256 identifier is d9529a61e677c3e16bda524bea9add508b8bb3b476d96ac3afd210fe4afe1512. A public access URL on https://github.com/d-kemolife/SASQOF-Security-Aware-Selective-Quantization-and-Optimization-Framework-, and the collection protocol on https://docs.google.com/document/d/1mD5i4nBtjqU1lntm2oowYqFyWIWPd1BV/edit?usp=sharing&ouid=113718009705312104100&rtpof=true&sd=true, have been provided. The supplementary archive in the github contains the corrective source code, split memberships, environment records, prediction files, model checkpoints, timing samples, and statistical summaries; it includes the raw input CSV. Independent rerunning requires the same source file and access through the URLs provided.

Declaration of competing interest

The authors declare no competing interests.

Funding

The authors report no specific funding for this study.

Acknowledgment

The authors would like to thank the supervisory team of the Department of Computer Science, Federal University Lokoja, Nigeria for their guidance.

References

[1] E. O. Oladipe, S. E. Adewumi, T. Kolajo & J. B. Agbogun, "Crop recommendation in precision agriculture: a systematic literature review of methods, trends, and challenges", African Scientific Reports **5** (2026) 590. https://doi.org/10.46481/asr.2026.5.3.590.

[2] K. Taueatsoala, C. Daniels, A. J. Ramsunar, P. Bronkhorst & A. E. Ezugwu, "TinyML-enabled IoT for sustainable precision irrigation", arXiv preprint (2026), arXiv:2601.13054. Available online: https://arxiv.org/abs/2601.13054.

[3] J. Lin, W.-M. Chen, Y. Lin, J. Cohn, C. Gan & S. Han, "MCUNet: Tiny deep learning on IoT devices", Advances in Neural Information Processing Systems **33** (2020) 11711. Available online: https://proceedings.neurips.cc/paper/2020/hash/86c51678350f656dcc7f490a43946ee5-Abstract.html.

[4] R. Bhaumik, B. Chakraborty, W. Choi, A. Dutta, J. Govinden & Y. Shen, "The committing security of MACs with applications to generic composition", in Advances in Cryptology—CRYPTO 2024, Springer, Cham, Switzerland, 2024, pp. 425–462. https://doi.org/10.1007/978-3-031-68385-5_14.

[5] G. Selander, J. Mattsson, F. Palombini & L. Seitz, "Object security for constrained RESTful environments (OSCORE)", Internet Engineering Task Force, RFC 8613 (2019). https://doi.org/10.17487/RFC8613.

[6] M. I. Akazue, "A fuzzy based intelligent irrigation system", SAU Science-Tech Journal **1** (2016) 12. Available online: https://www.researchgate.net/publication/370495372_A_fuzzy-based_intelligent_irrigation_system.

[7] H.-I. Liu, M. Galindo, H. Xie, L.-K. Wong, H.-H. Shuai, Y.-H. Li & W.-H. Cheng, "Lightweight deep learning for resource-constrained environments: a survey", ACM Computing Surveys **56** (2024) 267. https://doi.org/10.1145/3657282.

[8] B. Jacob, S. Kligys, B. Chen, M. Zhu, M. Tang, A. Howard, H. Adam & D. Kalenichenko, "Quantization and training of neural networks for efficient integer-arithmetic-only inference", Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, USA, 2018, pp. 2704–2713. https://doi.org/10.1109/CVPR.2018.00286.

[9] PyTorch Contributors, "quantize_dynamic", PyTorch API documentation. Available online: https://docs.pytorch.org/docs/stable/generated/torch.ao.quantization.quantize_dynamic.html (accessed 16 October 2026).

[10] PyTorch Contributors, "LSTM", Dynamic Quantized Modules, PyTorch API documentation. Available online: https://docs.pytorch.org/docs/stable/generated/torch.ao.nn.quantized.dynamic.LSTM.html (accessed 16 October 2026).

[11] J. Li & R. Alvarez, "On the quantization of recurrent neural networks", arXiv preprint (2021), arXiv:2101.05453. Available online: https://arxiv.org/abs/2101.05453.

[12] S. Maniam, Y.-K. Tee, E. Memar, H. Y. Wong & M. Zaman, "Smart irrigation management: IoT-based RNN-LSTM model for soil moisture prediction in precision agriculture", Smart Agricultural Technology **13** (2026) 101866. https://doi.org/10.1016/j.atech.2026.101866.

[13] S. Han, H. Mao & W. J. Dally, "Deep compression: compressing deep neural networks with pruning, trained quantization and Huffman coding", International Conference on Learning Representations, 2016. Available online: https://arxiv.org/abs/1510.00149.