Comparing Random Forest, XGBoost, LSTM, GRU, and Autoencoder Architectures for Cyber-Attack Detection in Industrial Control Systems

Five-Model Comparison on the SWaT Benchmark

Prerana Mahajan[#1], Aniket Lande[#2]

[#]*Zeal Institute of Business Administration, Computer Application and Research (ZIBACAR), Savitribai Phule Pune University, Pune, India*

[1]prernamahajan1805@gmail.com, [2]aniketlande1910@gmail.com

Abstract—

Cyber-physical attacks on industrial control systems (ICS) can create serious safety and operational hazards in critical infrastructure such as water treatment plants, power grids, manufacturing lines and transport networks. These plants stream sensor and actuator data around the clock, yet turning that data into reliable attack detection is difficult. Operating conditions drift over time, attacks occupy only a small share of the recorded runtime, and a detector that raises too many false alarms soon loses the trust of the operators who have to respond to it. This paper compares five learning approaches for separating normal plant operation from attack activity. Random Forest and eXtreme Gradient Boosting (XGBoost) serve as strong baselines for tabular features, Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) model temporal dependence, and an Autoencoder learns the bounds of normal behaviour and flags observations whose reconstruction error exceeds a threshold. The experiments use the Secure Water Treatment (SWaT) benchmark, which holds 1,387,098 normal records and 54,621 attack records across 51 physical variables. Models are judged by accuracy, precision, recall, F-score and confusion-matrix counts. On a held-out test set of 288,344 samples, Random Forest reached an accuracy of 99.9931 percent with 6 false positives and 14 false negatives, while XGBoost reached 99.9185 percent with 20 false positives and 215 false negatives. The LSTM currently has only a preliminary accuracy of 97.5 percent, and the full confusion matrices for the LSTM, GRU and Autoencoder are still being compiled. The paper also examines why the widely used 80/20 stratified random split is a weak basis for evaluating time-series models: consecutive one-second records are almost identical, so test samples effectively leak information from the training set and inflate the reported metrics. We outline the chronological and episode-isolated validation needed to confirm these findings.

Keywords—Industrial Control Systems; Cyber-Attack Detection; Anomaly Detection; Secure Water Treatment (SWaT); Random Forest; XGBoost; LSTM; GRU; Autoencoder.

I. INTRODUCTION

Operational Technology (OT) is the collection of industrial computers, specialised network protocols, field sensors, actuators and control logic that together monitor and run physical processes. A water treatment plant makes a convenient example. Instruments continuously record flow rates, pressure differences, tank levels and chemical indicators such as pH, dissolved oxygen and oxidation-reduction potential, while pumps driven by variable-frequency drives, motorised isolation valves and chemical dosing units carry out the plant's physical response. Programmable Logic Controllers (PLCs) and Distributed Control Systems (DCS) sit between the two, comparing each sensor reading with its setpoint and running the feedback loops that adjust the actuators. The same architecture, with different equipment, operates electrical transmission networks, petrochemical refineries, pharmaceutical batch plants and automated factories.

Protecting such systems is not the same task as protecting a corporate IT network. Enterprise security is organised around the CIA triad of confidentiality, integrity and availability, in that order of emphasis. In an industrial plant the priorities are effectively reversed: human safety, uninterrupted operation and the physical condition of the equipment come first. An unplanned host restart, aggressive in-line traffic inspection or a late blocking action can disturb a deterministic control loop and lead to equipment damage, environmental discharge or a risk to the surrounding community. For this reason, NIST Special Publication 800-82 Rev. 3 stresses that any cyber defence in OT must be weighed against plant availability, latency and mechanical stability [1].

These constraints change what a good anomaly detector looks like. Broad statistical accuracy is not enough. A detector intended for real use has to be judged on how quickly it reacts compared with the time the process needs to reach a damaging state, on how often it raises false alarms that wear down control-room staff, and on whether its output helps a field engineer locate the affected process stage.

A. Problem Statement

Process variables in a plant are tied together by physical laws, closed-loop actuator feedback, process transitions and external load, and they change at different rates. An attacker who manipulates a sensor signal, alters a setpoint or tampers with an actuator can do so in a way that produces only small, slow deviations, which are easily mistaken for ordinary operational drift.

Industrial telemetry is also heavily imbalanced. Normal operation fills almost the entire record, and attacks appear as rare, short interruptions. On data like this, a learning algorithm can reach a high overall accuracy simply by predicting the normal class every time, while missing the low-frequency attacks that matter most. Establishing the real trade-off between false alarms and missed events therefore needs a consistent benchmark that places supervised tabular ensembles and deep sequence models under the same conditions.

B. Aim

The aim of this study is to build, validate and critically assess a benchmarking framework that separates normal from attack states in industrial control systems using five model

families: Random Forest, XGBoost, LSTM, GRU and an Autoencoder.

C. Objectives

- To examine the physical and statistical properties of multi-sensor, multi-actuator telemetry that make attack detection difficult.
- To set up a reproducible data pipeline with documented timestamp handling, feature filtering, label harmonisation and strictly separated training, validation and test partitions.
- To train, tune and evaluate the five selected models on one standardised industrial dataset.
- To measure accuracy, precision, recall and F_1-score against exact confusion-matrix counts, reporting false positives (nuisance alerts) and false negatives (missed attacks) separately.
- To describe how sensitivity thresholds, false-alarm burden and detection reliability trade off against one another.
- To identify methodological weaknesses, in particular the temporal data leakage that random splitting causes, and to state the requirements for deployment in live plants.

D. Research Questions

- How do tree-based tabular ensembles compare with deep recurrent networks, in both classification performance and computational cost, on multivariate industrial telemetry?
- Does modelling temporal dependence explicitly with gated recurrent networks give a measurable advantage in detecting multi-stage attacks?
- How far does overall accuracy hide an unacceptable false-negative rate on the minority attack class in imbalanced industrial data?
- What validation standards (for example chronological partitioning, episode holdouts and latency measurement) must an anomaly detector meet before it can be called deployable?

II. Research Gap, Scope and Contributions

A. Research Gap

Class imbalance is a recognised weakness of both conventional anomaly detectors and supervised classifiers in industrial settings [9], [10]. A model tested on a heavily skewed split can score above 96% accuracy by labelling every input as normal, even though it detects none of the real intrusions. In critical infrastructure this outcome is unacceptable, because an unnoticed valve manipulation or tank overfill can destroy equipment.

A second weakness lies in how different model families are compared. Decision-tree ensembles split on individual feature thresholds, recurrent networks carry temporal progression in a latent state vector, and autoencoders learn the shape of normal behaviour by penalising reconstruction error. Published comparisons of these approaches often differ in preprocessing, feature representation and thresholding method, or they rely on leaky data partitions. A particularly common habit is to shuffle a continuous time series before splitting it into training and test sets. Because adjacent one-second records describe nearly the same physical state, many test samples then have almost exact copies in the training data, and the resulting metrics are inflated. This study responds to both problems with a controlled, like-for-like comparison of the five model families under shared preprocessing, explicit tracking of class imbalance and transparent evaluation criteria.

B. Research Focus

The study concerns binary attack identification in multivariate sensor and actuator streams, and compares the five architectures on accuracy, precision, recall and F_1-score. For the LSTM, only a baseline test accuracy of 97.5% is available so far. It is treated as an initial benchmark that still needs a full confusion matrix and time-aware validation against the saved test arrays.

C. Scope

The experiments are limited to binary classification, separating normal operation (Class 0) from attack-affected behaviour (Class 1), and all trials are run offline on labelled historical telemetry. Distinguishing between individual attack types, such as a single-stage sensor bias and a coordinated multi-stage attack, is left as a later diagnostic extension.

The models are not intended to attribute an intrusion, trace the root cause of a vulnerability or take automatic corrective action. Within a defence-in-depth arrangement they act only as decision support that complements the plant's existing monitoring.

D. Expected Contributions

- **Controlled multi-paradigm comparison.** A head-to-head evaluation of five classical and deep learning models on one identical industrial benchmark.
- **Separation of accuracy from operational metrics.** An analysis that treats global accuracy, minority-class recall and false-positive frequency as distinct quantities and relates classification errors to their physical consequences.
- **Leakage analysis.** A critique of how random sampling of continuous telemetry creates autocorrelation leakage, together with guidance on chronological and episode-isolated partitioning.
- **Deployment criteria.** A set of practical prerequisites, covering inference latency, alarm thresholds and operator interpretability, to be met before industrial use.

III. Background and Related Work

A. Industrial Control Systems and OT Security

A modern industrial control system is a layered structure. Field sensors and actuators connect to PLCs and Remote Terminal Units (RTUs), which report to SCADA servers and Human-Machine Interfaces (HMIs) over industrial buses such as Modbus, PROFINET and EtherNet/IP. Sensors turn physical quantities (fluid pressure, tank level, chemical conductivity) into electrical signals, PLCs run ladder logic or PID routines on those inputs, and the resulting commands go back to the actuators. Since these commands directly control the flow of energy and material, a compromised system can lead to environmental contamination, equipment destruction or loss of life.

NIST SP 800-82 Rev. 3 requires that security measures in OT preserve deterministic timing, mechanical availability and operational safety [1]. A deviation in process data does not always mean an attacker is present. Component wear, routine valve maintenance, sensor drift and sudden changes in demand all create statistical anomalies that resemble attacks. Data-driven detectors should therefore be read as investigative indicators, to be checked against the physical process context, network packet records and the knowledge of experienced operators.

B. Industrial Time-Series Characteristics

Industrial telemetry consists of strictly ordered measurements taken at a fixed interval, typically 1 Hz. Physical equipment has mechanical and thermal inertia, so consecutive samples are strongly autocorrelated. Switching off a pump, for example, changes downstream flow and upstream pressure gradually over several seconds rather than in one step.

This continuity has a direct effect on validation. If individual records are assigned to training and test folds at random, the predecessors and successors of many test records sit in the training fold. A model evaluated this way can score well by memorising operating trajectories instead of learning general process dynamics. Evidence of real generalisation requires a strict chronological split or a scheme that holds out entire attack episodes. Fig. 1 contrasts the two partitioning approaches.

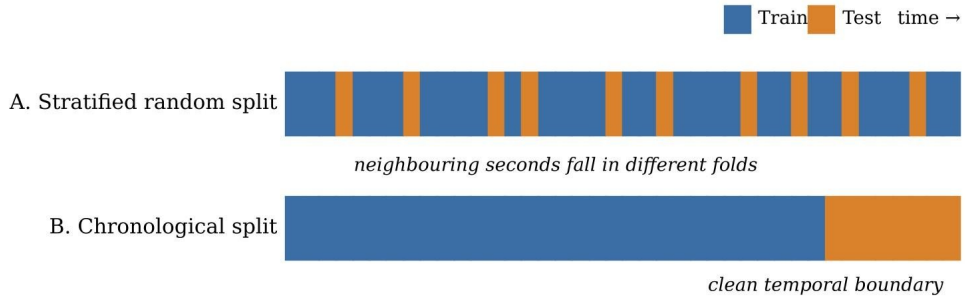

Fig. 1. Random (stratified) splitting scatters neighbouring one-second records across the training and test folds, whereas a chronological split keeps a clean temporal boundary between them.

The Secure Water Treatment (SWaT) testbed, built by iTrust at the Singapore University of Technology and Design (SUTD), is widely used as a reference platform for cyber-physical security research [2]. It reproduces a six-stage municipal water purification process and records sensor and actuator logs during both normal operation and scripted multi-stage attacks [3]. Later studies have used SWaT to build behavioural baselines and correlation-based anomaly detection frameworks [9], [10].

C. Tree-Based Ensemble Architectures

A Random Forest grows many decorrelated decision trees, each fitted to a bootstrap sample of the training data with a random subset of features considered at every split. Predictions are combined by majority vote, which reduces the variance of any single tree and allows the forest to capture non-linear feature interactions without feature normalisation. Like other standard tree models, however, it assumes independent and identically distributed (i.i.d.) observations and ignores the order of samples unless lagged temporal features are added to the input.

XGBoost is a sequential gradient-boosting method that minimises a regularised objective made up of a convex loss and a penalty on tree complexity [4]. Each new tree is fitted to the negative gradient of the loss with respect to the current ensemble output. XGBoost performs very well on tabular data and supports cost-sensitive class weighting, but like Random Forest it has no sense of chronological order unless sequential features are engineered for it.

D. Recurrent Architectures: LSTM and GRU

Long Short-Term Memory networks address the vanishing-gradient problem of plain recurrent networks by adding a memory cell c_t controlled by three multiplicative gates: a forget gate f_t, an input gate i_t and an output gate o_t [5]. The update at each time step is:

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \quad (1)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (2)$$

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (3)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (4)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (5)$$

$$h_t = o_t \odot \tanh(c_t) \quad (6)$$

Here σ is the logistic sigmoid, W and b are learned weights and biases, $[h_{t-1}, x_t]$ is the concatenation of the previous hidden state with the current input, and the circled-dot operator denotes element-wise multiplication. Through this gated memory an LSTM can follow a physical trajectory over a chosen time horizon and tell apart a persistent multivariate deviation, which may indicate an attack, from brief process noise. Gated Recurrent Units simplify the structure by merging the cell state and hidden state into a single vector h_t, governed by an update gate z_t and a reset gate r_t [6]:

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t] + b_z) \quad (7)$$

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t] + b_r) \quad (8)$$

$$\tilde{h}_t = \tanh(W \cdot [r_t \odot h_{t-1}, x_t] + b) \quad (9)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \quad (10)$$

Because a GRU has fewer trainable parameters, it often matches the sequence-modelling ability of an LSTM while training faster and running inference more quickly.

E. Autoencoder-Based Unsupervised Anomaly Detection

An autoencoder has two parts. The encoder $g_\varphi(x)$ maps an input vector x in R^D to a latent vector z in R^d of much lower dimension (d is far smaller than D), and the decoder $f_\theta(z)$ maps z back to the input space, giving the reconstruction $\hat{x} = f_\theta(g_\varphi(x))$. When the network is trained only on normal operating data by minimising a reconstruction loss such as the mean squared error (MSE),

$$\mathcal{L}_{MSE}(x, \hat{x}) = \frac{1}{D}\sum_{j=1}^{D}(x_j - \hat{x}_j)^2 \quad (11)$$

its weights come to encode the physical correlations of uncompromised operation. An attack record, in which those correlations are disturbed, then tends to produce a larger reconstruction error. A threshold τ, set on normal validation data, converts the error into a decision:

$$\hat{y} = 1 \text{ ιφ } \mathcal{L}_{MSE}(x, \hat{x}) > \tau, \quad \hat{y} = 0 \text{ οτηερωισε} \quad (12)$$

This idea is related to other unsupervised boundary-based approaches such as Isolation Forests [7] and variational latent models [8]. Unlike a supervised classifier, an autoencoder is a one-class novelty detector, so its usefulness depends almost entirely on calibrating the threshold carefully against validation data.

F. Positioning Within the Literature

The literature shows that both classical ensembles and deep neural networks can detect abnormal conditions in industrial telemetry, but no single architecture performs best in every operating regime. This study adds to that body of work by fixing a common baseline: five model families are evaluated with identical feature criteria, the weakness of global accuracy as a measure is made explicit, and the leakage problem in time-series validation is addressed directly.

IV. DATASET AND DATA DESCRIPTION

A. Dataset Identification

The experiments use the Secure Water Treatment (SWaT) dataset released by iTrust at SUTD [2], [3]. The data comes in three files: normal.csv, attack.csv and a combined sequence, merged.csv. Table I summarises the record counts, feature dimensions and time span.

TABLE I. DATASET SUMMARY AND PHYSICAL PARAMETERS

Attribute	Specification
Dataset name	Secure Water Treatment (SWaT), December 2015 release (SWaT.A1)
Originating facility	iTrust, Singapore University of Technology and Design; Goh et al. [3]
Normal records	1,387,098 continuous operational observations
Attack records	54,621 active cyber-physical attack observations
Total records	1,441,719 observations
Feature dimensions	51 continuous variables (26 physical sensors, 25 mechanical actuators)
Sampling frequency	1.0 Hz (one observation per second)
Binary label	Class 0: normal operation; Class 1: cyber-attack incident
Temporal coverage	22 December 2015, 16:00:00 to 02 January 2016, 14:59:59 (per timestamps)

B. Features, Actuator Signals and Data Hygiene

The 51 attributes describe the six purification stages of the plant: raw water intake (P1), chemical pre-treatment and coagulation (P2), ultrafiltration membrane separation (P3), dechlorination (P4), reverse osmosis (P5) and clean water distribution (P6). The features are continuous analogue sensor readings (level transmitters LIT, flow transmitters FIT, pH meters and conductivity analysers) and discrete actuator states (motorised valves MV and booster pumps P). Table II lists example instruments from each stage.

TABLE II. PROCESS STAGES OF THE SWAT PLANT AND EXAMPLE INSTRUMENTS

Stage	Process	Example instruments
P1	Raw water intake and tank level	FIT101, LIT101
P2	Chemical dosing and coagulation	AIT201-203, P201
P3	Ultrafiltration membrane separation	LIT301, DPIT301
P4	Dechlorination and UV disinfection	AIT401, P401
P5	Reverse osmosis, high-pressure system	FIT501-504
P6	Clean water storage and backwash	LIT601, P601

The original categorical and string labels were converted to a binary target y in {0, 1}. Timestamps were parsed so that records could be ordered and windowed, but they were kept out of the feature matrix on purpose; otherwise a model could learn artefacts of the calendar rather than the physical dynamics. Missing numerical values were filled with column-wise medians computed on the training data, and rows containing infinite or non-numeric values were removed.

C. Class Imbalance

Attack records make up only 3.788% of the data (54,621 of 1,441,719 samples), which is a nominal-to-attack ratio of 25.4:1. A trivial classifier that predicts normal operation for every record would therefore score 96.212% accuracy while detecting none of the attacks. For this reason, accuracy cannot serve as a standalone evaluation measure here. Where cost-sensitive weighting, undersampling and synthetic augmentation (for example SMOTE) were tried, they were applied only inside the training pipeline so that no synthetic data could contaminate the other partitions.

D. Data Quality Assurance

- **Source verification.** Checksums, row boundaries and label mappings in normal.csv and attack.csv were checked to confirm where the data came from.
- **Temporal alignment.** A uniform one-second sampling interval was verified, and missing time steps and frozen-sensor periods were identified.
- **Range checking.** Process values were compared with realistic operating ranges to catch corrupted analogue readings and sensor saturation.

- **Leakage prevention.** Scaling, median imputation and normalisation statistics were fitted on the training partition only and then applied unchanged to the validation and test data.

V. METHODOLOGY

A. Experimental Workflow

The workflow consists of data verification, cleaning, train/validation/test partitioning, model training, threshold optimisation and held-out evaluation. To keep the comparison fair, every model was tested on the same held-out split of 288,344 samples. Fig. 2 gives an overview, and the steps are described below.

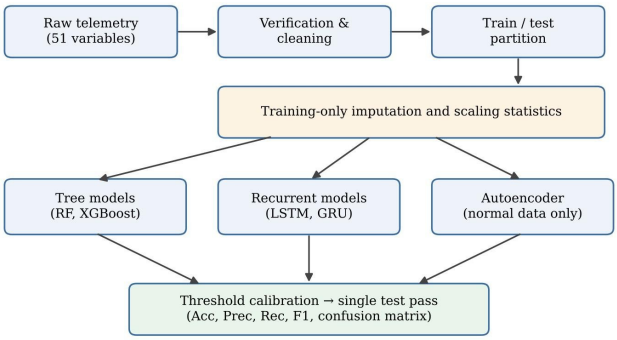

Fig. 2. Overview of the experimental pipeline. Imputation and scaling statistics are fitted on the training partition only.

- **Data ingestion and integrity checks.** Confirm the dataset structure, the 51 telemetry channels, the binary label encoding and the time limits.
- **Exploratory analysis.** Examine multivariate correlations, class distribution and physical state transitions across stages P1 to P6.
- **Cleaning and preprocessing.** Impute missing values with training medians, set timestamps aside, and fit the standard scalers on training data only.
- **Partitioning.** Create an 80/20 stratified train/test split as the baseline, and record the temporal-leakage risk that comes with it.
- **Model configuration and training.** Train the Random Forest, XGBoost, LSTM, GRU and Autoencoder with the chosen regularisation and optimisation settings.
- **Threshold calibration.** Choose the decision cutoffs of the supervised models on validation data, and set the Autoencoder reconstruction threshold τ from percentiles of the normal validation error.
- **Held-out inference.** Run a single evaluation pass over the unseen test partition of 288,344 samples and save the prediction vectors.
- **Metric computation.** Calculate the exact confusion matrices, accuracy, precision, recall and F_1-scores.
- **Physical error analysis.** Interpret the false positives and false negatives in terms of actual plant operation.

B. Random Forest

The Random Forest was trained on the 51-dimensional tabular features. The main hyperparameters were the number of trees (B = 100), the maximum tree depth, the minimum number of samples required to split a node, and the size of the feature subset considered at each split (m = sqrt(D)). Gini-based feature importances show which telemetry channels best separate normal from compromised states, but they reflect predictive correlation and do not establish that a channel physically causes an attack.

C. XGBoost

The XGBoost model builds boosted trees using a second-order Taylor expansion of the loss [4]. The tuned settings included the number of boosting rounds, the shrinkage rate (η), maximum depth, row and column subsampling ratios, and the L1 and L2 regularisation terms (α and λ). To offset the class imbalance, scale_pos_weight was set according to the ratio of normal to attack records. Early stopping on the validation loss was used to limit overfitting at the leaf level.

D. LSTM

The LSTM was set up for multivariate time-series classification with fixed-length sliding windows. With F = 51 features and a window of T consecutive observations, the input tensor has shape (N, T, F), where N is the number of windows, and each window takes the label of its last time step. Windows were built within partition boundaries so that no window spans the training and test data. Features were standardised with a z-score scaler fitted on the training partition only:

$$z = \frac{x - \mu_{train}}{\sigma_{train}} \tag{13}$$

The network stacks several LSTM layers with dropout (p = 0.2) between them to reduce co-adaptation, followed by a fully connected output layer with a sigmoid activation for binary prediction. Training minimised binary cross-entropy with the Adam optimiser.

E. GRU

The GRU model reuses the LSTM's preprocessing, window shape (N, T, F) and normalisation so that the two can be compared directly. Because its reset and update gates share one hidden vector, it has fewer parameters than the LSTM, needs less memory and completes each training epoch faster. The evaluation therefore looks mainly at how much detection sensitivity is gained or lost relative to the saving in runtime cost.

F. Autoencoder

The Autoencoder was built as an unsupervised novelty detector and trained only on normal records (y = 0). Its encoder compresses the 51-dimensional input through dense layers with non-linear activations (for example ReLU) into a lower-dimensional bottleneck z, and its decoder expands z back to a 51-dimensional reconstruction. Training minimises the MSE on normal data.

At test time, records with out-of-range actuator positions or broken sensor correlations are expected to give higher reconstruction errors. The threshold τ was calibrated on a validation set of normal samples to match a chosen false-alarm budget, for example the 99.5th percentile of normal validation error. By construction, such a rule accepts roughly 0.5% of normal validation records as false alarms. On the 277,420 normal records in the test set, a similar rate would amount to well over a thousand false alarms, so the choice of percentile

will weigh heavily on the Autoencoder's precision once its results are final.

VI. Evaluation Metrics

A. Confusion-Matrix Outcomes

Every prediction in binary attack classification falls into one of four outcomes:

- **True Positive (TP):** an active attack is correctly flagged.
- **True Negative (TN):** normal operation passes without an alert.
- **False Positive (FP):** normal operation is wrongly flagged as an attack (a false alarm or nuisance trip).
- **False Negative (FN):** an active attack goes unflagged (a missed intrusion).

All four counts must be examined together, since aggregate scores on their own can conceal serious operational risks.

B. Accuracy

Accuracy is the share of all predictions that are correct:

$$\text{Αχχυραχψ} = \frac{TP+TN}{TP+TN+FP+FN} \tag{14}$$

Because normal samples dominate the data, high accuracy is only a sanity check here and says little about how well attacks are detected.

C. Precision

Precision indicates how far an alert can be trusted:

$$\Pi\rho\varepsilon\chi\iota\sigma\iota\sigma\nu = \frac{TP}{TP+FP} \tag{15}$$

In a working plant, high precision means that alarms usually correspond to real incidents, so operators are less likely to dismiss them out of alarm fatigue. If a model raises no alarms at all (TP + FP = 0), precision is undefined and is conventionally reported as zero under strict evaluation rules.

D. Recall (Sensitivity)

Recall measures how many of the real attacks the model finds:

$$\text{Ρεχαλλ} = \frac{TP}{TP+FN} \tag{16}$$

For industrial control systems recall is the main safety metric, because each false negative is an attack that continues unnoticed inside the physical process.

E. F_1-Score

The F_1-score is the harmonic mean of precision and recall, so it balances the credibility of alerts against the coverage of detection:

$$F_1 = 2 \cdot \frac{\Pi\rho\varepsilon\chi\iota\sigma\iota\sigma\nu \cdot \text{Ρεχαλλ}}{\Pi\rho\varepsilon\chi\iota\sigma\iota\sigma\nu + \text{Ρεχαλλ}} = \frac{2TP}{2TP+FP+FN} \tag{17}$$

Since true negatives do not appear in the formula, the F_1-score reflects performance on the minority attack class and is not inflated by the large number of normal records.

F. Additional Metrics

Table III lists the primary and secondary measures used in the benchmark. For imbalanced data it is also advisable to report the area under the precision-recall curve (PR-AUC), which summarises behaviour across all decision thresholds.

TABLE III. Evaluation Metrics and Their Operational Interpretation

Metric	Formulation	Interpretation in industrial control systems
Accuracy	(TP + TN) / (TP + TN + FP + FN)	Overall correctness; strongly affected by class imbalance and can hide high miss rates.
Precision	TP / (TP + FP)	Share of raised alerts that are genuine attacks; guards against operator alarm fatigue.
Recall (sensitivity)	TP / (TP + FN)	Share of real attacks detected; critical for limiting physical damage.
F_1-score	2 × Precision × Recall / (Precision + Recall)	Balance of alert credibility and detection coverage on the attack class.
False-positive rate	FP / (FP + TN)	Fraction of normal states that trigger false alarms; sets control-room workload.
Mean detection delay	$t_{alert} - t_{onset}$	Time between attack onset and the first alert; must stay inside safety margins.

VII. Experimental Design and Reproducibility

A. Train, Validation and Test Partitioning

The baseline configuration uses an 80/20 train/test split drawn by stratified random sampling (test_size = 0.2, random_state = 42).

This design carries a methodological risk, which we call temporal autocorrelation leakage. Because the plant is sampled at 1 Hz, its physical state changes smoothly from one second to the next. In a stratified random split, the records at times t−1 and t+1 can land in the training set while the record at time t goes to the test set. The plant state at time t is almost the same as that of its neighbours, so a model can classify the test record by simple local interpolation without understanding the underlying physics.

Metrics obtained under random splitting should therefore be read as an optimistic upper bound. A definitive validation needs either a strict chronological split, which trains on earlier operating periods and tests on later ones, or an attack-episode holdout, which reserves entire attack scenarios for testing. For the LSTM and GRU, sliding windows must also be built strictly inside each partition so that overlapping windows cannot carry information across the boundary.

B. Hyperparameter Tuning and Overfitting Safeguards

Hyperparameters should be tuned on inner validation splits or with rolling-origin time-series cross-validation, and the held-out test partition should be used exactly once, on the final trained models. The neural networks used early stopping, weight decay and dropout (p = 0.2), and the tree ensembles were constrained by a maximum depth and leaf regularisation to limit overfitting.

 ISSN: 3139-1141

C. Benchmarking Protocol

- **Uniform target definition.** Every model solves the same binary task on the same 288,344-sample test split.
- **Preprocessing isolation.** Imputation parameters and normalisation coefficients (mean and standard deviation) are computed on training data only.
- **Independent threshold tuning.** Decision thresholds and reconstruction limits are chosen on validation data before the test run.
- **Prediction archiving.** Probability arrays and class predictions are saved to disk so that the confusion matrices can be verified independently.

D. Reproducibility Checklist

Table IV records the experimental settings, environment details and the current status of the results, to support independent reproduction.

TABLE IV. EXPERIMENTAL REPRODUCIBILITY CHECKLIST

Parameter	Specification
Dataset release	Secure Water Treatment (SWaT.A1), Dec 2015; iTrust, SUTD [3].
Feature set	51 continuous variables (excluding timestamp and the binary normal/attack label).
Data cleaning	Non-numeric artefacts dropped; column medians imputed for missing values.
Partition	80% train, 20% held-out test (random_state = 42, stratified).
Class distribution, train	1,153,375 records (normal: 1,109,678; attack: 43,697).
Class distribution, test	288,344 records (normal: 277,420; attack: 10,924).
Model architectures	Random Forest, XGBoost, LSTM, GRU, Autoencoder.
Supervised decision cutoff	Standard probability threshold (0.50).
Autoencoder threshold rule	99.5th percentile of normal validation reconstruction error.
Status of confusion matrices	Recorded for Random Forest and XGBoost; pending final validation for LSTM, GRU and Autoencoder.

VIII. RESULTS AND DISCUSSION

A. Measured Results

Table V compares the models on the held-out test partition of 288,344 records, of which 277,420 are normal and 10,924 are attacks. The accuracy, precision, recall, F_1-score and error counts for Random Forest and XGBoost are verified test outcomes. For the LSTM only a preliminary accuracy of 97.5% has been reported. The remaining LSTM entries, and all GRU and Autoencoder entries, are marked [Measure] or [Count] until their confusion matrices have been verified.

TABLE V. MODEL PERFORMANCE ON THE HELD-OUT TEST PARTITION (288,344 SAMPLES)

Model	Acc. (%)	Prec. (%)	Recall (%)	F_1 (%)	FP	FN
Random Forest	99.9931	99.9450	99.8718	99.9084	6	14
XGBoost	99.9185	99.8136	98.0319	98.9147	20	215

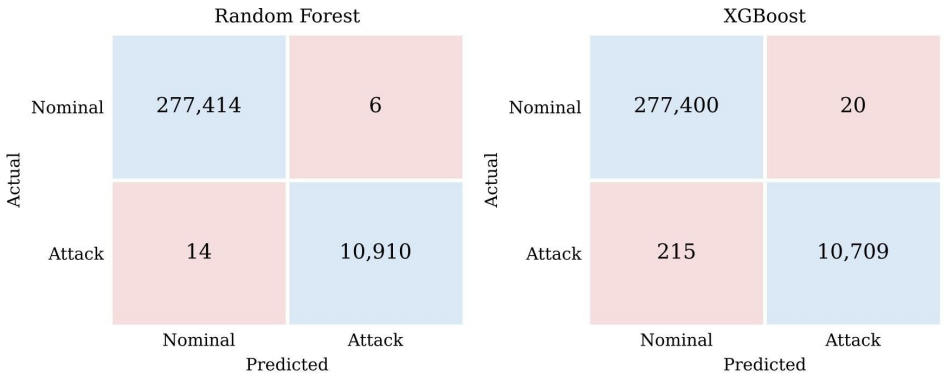

Fig. 3. Confusion matrices of the Random Forest and XGBoost models on the test set (N = 288,344). Cells show record counts.

B. Tree-Based Ensembles

Random Forest gave the strongest measured result on this partition: 99.9931% accuracy, 99.9450% precision, 99.8718% recall and an F_1-score of 99.9084%. It raised only 6 false alarms among the 277,420 normal test records (a false-positive rate of 0.0022%) and missed 14 of the 10,924 attack records. XGBoost reached 99.9185% accuracy, 99.8136% precision and an F_1-score of 98.9147%. Its recall, however, was lower at 98.0319%, which left 215 false negatives, more than fifteen times as many as Random Forest. At the 1 Hz sampling rate, those 215 missed records correspond to more than three and a half minutes of attack activity that went unflagged in a water plant.

Even though these figures are very strong, they need to be read with care. The test partition was drawn by stratified random sampling from 1 Hz telemetry, so neighbouring samples in the training fold probably helped both tree models. Validating them on a chronological holdout is therefore a main objective of the continuing work.

C. The Reported LSTM Result

The LSTM has a reported baseline accuracy of 97.5%. In many machine-learning problems that would be considered a good result, but on this imbalanced dataset it illustrates exactly why global accuracy is a poor guide. As Section IV-C showed, a classifier that never predicts an attack would already score 96.212% on the full dataset.

On 288,344 test records, an accuracy of 97.5% means roughly 7,209 misclassified observations (288,344 × 0.025). Random Forest made 20 errors in total (FP = 6, FN = 14) and XGBoost made 235 (FP = 20, FN = 215), as Fig. 4 shows. Since the attack class has only 10,924 records, an error count of

7,209 could, in principle, mean that the model misses a large majority of attacks if the errors are all false negatives. If instead they are false alarms, roughly 7,200 spurious alerts during normal operation would overwhelm plant operators and erode their trust in the system. The LSTM result therefore cannot be judged until its exact confusion matrix is available.

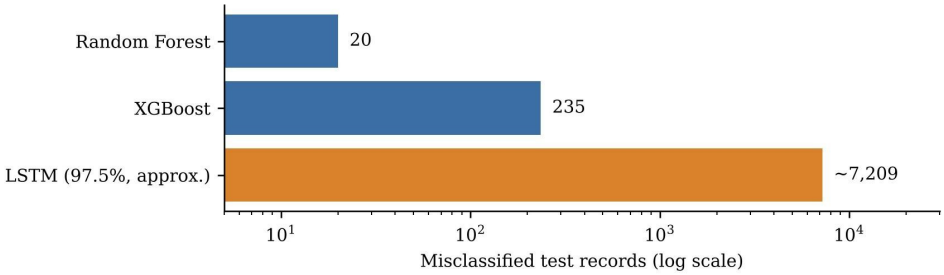

Fig. 4. Total misclassified test records per model (log scale). The LSTM bar is approximate and is derived from its reported 97.5% accuracy; its error split is unknown.

If the completed matrices show high recall on long, complex attacks that the tree baselines fail to detect, that would be clear evidence in favour of sequence modelling. If, on the other hand, training accuracy turns out to be much higher than validation accuracy, the network may be overparameterised for tabular sensor inputs, which would call for a smaller architecture or stronger regularisation.

D. Comparison of the Architectures

The five model classes differ in both operational behaviour and computational demands.

- **Random Forest** handles multi-rate tabular features well without much input scaling and produced few false alarms. It has no built-in way to represent temporal trends unless lagged variables are supplied.
- **XGBoost** is effective on structured data and offers adjustable regularisation and class weights. It needs careful tuning, since poorly chosen early stopping or depth limits can reduce minority-class recall, as the 215 false negatives here suggest.
- **LSTM and GRU** are designed to capture temporal dynamics and may therefore detect slow-drift attacks that static models overlook. They need more training compute, more hyperparameter search and carefully designed sliding windows.
- **Autoencoder** works as an unsupervised one-class detector and does not require attack labels. Its reliability depends heavily on the reconstruction threshold τ, and it can be unsettled by harmless operating transitions and instrument noise.

Choosing a model for deployment means weighing safety risk against the staff available to respond. Where missing an attack is the overriding concern, the model should be tuned for maximum recall and fast detection. A facility with few operators may prefer high precision and a low false-positive rate to avoid alert fatigue. In either case the decision should rest on these site-specific trade-offs and not on headline accuracy.

E. Error Analysis

A first look at the misclassifications suggests several questions for a targeted error analysis.

- **False negatives (missed attacks).** Do the undetected attack samples cluster in particular stages, such as the ultrafiltration backwash cycle, or do they occur in multi-stage attacks where the physical variables stay inside normal ranges?
- **False positives (nuisance alerts).** Are false alarms triggered by routine transitions such as raw-water pump start-ups or scheduled chemical dosing changes?
- **Feature engineering.** Would temporal derivative features (the change in x per unit time) or known physical invariants, such as the mass balance linking inflow FIT101 to tank level LIT101, help resolve borderline cases?

F. Statistical Significance and Correlated Samples

Because 1 Hz telemetry is strongly autocorrelated, confidence intervals that assume independent Bernoulli trials will understate the true variance. Confidence bounds for accuracy, precision and recall should instead use moving-block bootstrap methods or non-overlapping block cross-validation to keep the statistical inference valid.

IX. Limitations, Risks and Operational Considerations

A. Representativeness of the Dataset

SWaT is a realistic emulation of a municipal water purification plant, but it is still an experimental facility run under controlled conditions. Real plants face seasonal changes in source water, unscripted maintenance, ageing components and varying sensor drift. A model trained only on SWaT cannot be moved to another facility without domain adaptation, plant-specific retraining and recalibration of its baseline.

B. Ground-Truth Labelling and Physical Effects

The binary SWaT labels mark the intervals in which attacks were launched in the experiments. A cyber attack, however, may not change the physical telemetry at the moment it is executed, because fluid transit times, pipe volumes and mechanical inertia delay its effect. After an attack ends, the process variables may take minutes to settle back to their setpoints. Treating the attack interval as strictly binary ignores these transition periods and may cause recovery phases to be counted wrongly as false alarms or missed detections.

C. False Alarms, Missed Events and Alert Fatigue

In a working control room, too many false alarms lead operators to silence or ignore the monitoring system, while false negatives let an attacker manipulate the process

unchecked. Industrial deployments have to balance the two by setting thresholds that keep the false-alarm rate manageable, and by keeping physical safety limits in place through independent hardwired interlocks. Table VI summarises the consequences of each error type.

TABLE VI. OPERATIONAL IMPACT OF CLASSIFICATION ERRORS

False positive (FP)	False negative (FN)
Operator alarm fatigue	Adversarial manipulation of the physical process
Possible unnecessary plant shutdowns	Chemical contamination of treated water
High cost of investigating alerts	Safety and regulatory-compliance hazards

D. Deployment and Safety Constraints

The models studied here are offline diagnostic detectors and are not certified for autonomous actuation in safety-critical systems. Their outputs must never override primary Safety Instrumented Systems (SIS) or hardwired electromechanical interlocks. In a real facility they should run passively, outside the control path, and feed a secondary alarm display for engineers to review, while the plant itself remains under deterministic control logic.

E. Interpretability and Model Lifecycle

Operators need clear, usable explanations when a model reports an anomaly. Deep neural networks and large tree ensembles do not explain themselves in physical terms. Feature attribution methods such as SHAP (Shapley Additive exPlanations) can show which sensors or actuators contributed most to an alert and so help technicians diagnose it.

Plants also change over time through maintenance, pipe descaling, pump replacement and seasonal shifts in water quality. These changes cause concept drift, so deployed models need continuous monitoring, automated data validation and a defined retraining procedure.

X. CONCLUSION AND FUTURE WORK

A. Conclusion

This study compared five machine learning and deep learning architectures (Random Forest, XGBoost, LSTM, GRU and an unsupervised Autoencoder) for detecting cyber attacks in industrial control systems, using the SWaT benchmark.

On an 80/20 stratified test partition of 288,344 samples, Random Forest gave the strongest measured result, with 99.9931% accuracy, 99.9450% precision, 99.8718% recall and an F_1-score of 99.9084%, along with 6 false alarms and 14 missed attack records. XGBoost reached 99.9185% accuracy and an F_1-score of 98.9147% but produced 215 false negatives, a miss rate of 1.968%. The LSTM has a reported accuracy of 97.5%, which corresponds to more than 7,200 misclassified records and shows why global accuracy cannot be relied on in isolation for imbalanced industrial data.

A central methodological finding is that stratified random sampling of continuous time-series telemetry introduces temporal autocorrelation leakage and so gives optimistic performance estimates. Time-aware validation is needed before any of these models can be considered operationally viable. In practice, an effective detector must combine high recall with a manageable false-alarm rate and low inference latency, and cannot be judged on headline accuracy alone.

B. Future Work

- **Temporal and episode-based validation.** Re-evaluate all five architectures under strict chronological holdout splits and leave-one-attack-episode-out cross-validation to remove temporal autocorrelation leakage.
- **Complete recurrent and generative results.** Finalise and verify the confusion matrices, precision, recall and F_1-scores for the LSTM, GRU and Autoencoder.
- **Attack-specific evaluation.** Break recall down by attack category to find which manipulation strategies evade detection.
- **Physical invariants.** Bring in known physical principles, such as differential equations for tank volume and mass-conservation balances, as input features or loss constraints.
- **Computational benchmarking.** Measure per-sample inference latency, memory footprint and power use on industrial edge hardware (for example Siemens IPCs and NVIDIA Jetson devices).
- **Explainable AI.** Add local SHAP and gradient-based attribution so that operators receive real-time explanations of raised alarms.

REFERENCES

[1] K. Stouffer, M. Pease, C. Tang, T. Zimmerman, V. Pillitteri, A. Hahn, S. Saravia, A. Sherule, and M. Thompson, "Guide to Operational Technology (OT) Security," NIST Special Publication 800-82 Rev. 3, National Institute of Standards and Technology, Gaithersburg, MD, Sep. 2023. doi: 10.6028/NIST.SP.800-82r3.

[2] A. P. Mathur and N. O. Tippenhauer, "SWaT: A Water Treatment Testbed for Research and Training on ICS Security," in Proc. 2016 International Workshop on Cyber-physical Systems for Smart Water Networks (CySWater), Vienna, Austria, 2016, pp. 31-36. doi: 10.1109/CySWater.2016.7469060.

[3] J. Goh, S. Adepu, K. N. Junejo, and A. Mathur, "A Dataset to Support Research in the Design of Secure Water Treatment Systems," in Critical Information Infrastructures Security (CRITIS 2016), Lecture Notes in Computer Science, vol. 10242, E. Rome, R. Bloomfield, and G. Spanoudakis, Eds. Cham: Springer, 2017, pp. 88-99. doi: 10.1007/978-3-319-71368-7_8.

[4] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in Proc. 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '16), San Francisco, CA, USA, 2016, pp. 785-794. doi: 10.1145/2939672.2939785.

[5] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," Neural Computation, vol. 9, no. 8, pp. 1735-1780, Nov. 1997. doi: 10.1162/neco.1997.9.8.1735.

[6] K. Cho, B. van Merriënboer, C. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," in Proc. 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Doha, Qatar, 2014, pp. 1724-1734. doi: 10.3115/v1/D14-1179.

[7] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in Proc. 2008 Eighth IEEE International Conference on Data Mining (ICDM), Pisa, Italy, 2008, pp. 413-422. doi: 10.1109/ICDM.2008.17.

[8] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in Proc. 2nd International Conference on Learning Representations (ICLR 2014), Banff, Canada, 2014. arXiv: 1312.6114.

[9] E. Birihanu and I. Lendák, "Explainable correlation-based anomaly detection for Industrial Control Systems," Frontiers in Artificial Intelligence, vol. 7, p. 1508821, Feb. 2025. doi: 10.3389/frai.2024.1508821.

[10] S. Ayas, M. S. Ayas, B. Çavdar, and A. K. Şahin, "Detecting cyberattacks based on deep neural network approaches in industrial control systems," Journal of Information Security and Applications, vol. 94, p. 104206, Nov. 2025. doi: 10.1016/j.jisa.2025.104206.