Accuracy-latency Trade-offs in Machine and Deep Learning Models for Real-time Smart Grid Intrusion Detection

J. E. Efiong[1,*], A. Akinwale[1], E. A. Olajubu[1], G. A. Aderounmu[2]

[1] Department of Computer Science and Cybersecurity, Obafemi Awolowo University, Ile-Ife, Nigeria.
[2] Department of Computer Engineering, Obafemi Awolowo University, Ile-Ife, Nigeria.

* Corresponding author. Email: jeefiong@oauife.edu.ng (J.E.E.); logistronics@yahoo.com (A.A.); emmolajubu@oauife.edu.ng (E.A.); gaderoun@oauife.edu.ng (G.A.A.)
Manuscript received August 29, 2026; accepted September 22, 2026; published November 20, 2026.
doi: 10.12720/sgce.15.2.108-122

Abstract: Real-time smart-grid intrusion detection requires models that are fast enough to keep up with operational traffic and accurate enough to safeguard vital control functions. This paper analyses aggregate results for nine intrusion-detection models from one smart-grid cyber range. It is a structured secondary analysis, not a new experiment or a statistical meta-analysis. The reported accuracies range from 98.32% to 100%, and the displayed model-inference times range from zero (after rounding) to 0.056879 ms. The study reproduces simple latency ratios and examines possible three-stage routing scenarios. The accuracy-latency trade-offs of these nine machine-learning and deep learning models, namely, Random Forest (RF), K-Nearest Neighbors (KNN), Extreme Gradient Boosting (XGBoost), Gated Recurrent Unit (GRU), Deep Belief Network (DBN), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Deep Neural Network (DNN), and Long Short-Term Memory (LSTM) were evaluated in an emulated multilayer cyber-physical smart grid intrusion detection system. Among positive-latency models, XGBoost and DNN form the principal non-dominated accuracy-latency frontier: XGBoost reports 99.9% accuracy at 0.000671 ms, while DNN reports 99.49% at 0.001156 ms and the best Layer 3 error profile. The paper identifies what can be inferred from the available tables and specifies the data and validation needed before any deployment recommendation is justified.

Keywords: cyber-physical systems, cybersecurity, deep learning, machine learning, smart grid, intrusion detection, pareto frontier, real-time systems

1. Introduction

Modern smart grids couple physical energy delivery with pervasive, bidirectional communication among intelligent electronic devices and protection relays, programmable logic controllers, supervisory control and data acquisition systems, field equipment, control centres, and remote operators [1, 2]. Wide-area connectivity enables network-based monitoring, automation, and control of operations. That same connectivity exposes operations to greater numbers of intrusion attempts on compromised datasets and commands. Those intrusion attempts can affect systems integral to power-grid management, where time plays a crucial role [1, 3–5]. The operation of a detector within this scenario presents two simultaneous challenges. First, it needs to discriminate reliably between legitimate and harmful network activity, controlling false-positive and false-negative errors [6, 7]. Second, it needs to return predictions in a timely manner that matches flow dynamics and the schedule of controllers. A detector that meets its classification

objectives may introduce unacceptable latencies. Those latencies can queue messages or delay countermeasures. High speeds do little good when they create unacceptable processing loads or overlook harmful events. Accuracy alone thus cannot govern architectural decisions. The choice among machine-learning and deep-learning paradigms falls within the trade-off between accuracy, latency, and costs. Decision tree ensembles include Random Forest and Extreme Gradient Boosting [8].

They evaluate predictions using traversal through ensembles of trees. This inference approach enables non-linear classification or regression decisions that depend largely on ensemble cardinality and tree depth. KNN is an instance-based model. It keeps its training data and compares distances to classify a new sample. Its computational cost depends on the cardinality of its reference set and the dimensionality of its input. Feed-forward DNNs and CNNs encode feature relationships across inputs, while sequence-based RNNs, GRUs, and LSTMs use recurrence and gates to encode temporal dependencies [9–13].

There have been significant advances in the smart-grid intrusion detection literature. Classification metrics including accuracy, precision, recall, and F1-Scores remain important measures. Yet inference times rarely receive attention outside standalone numbers, which creates several problems. Is an incrementally better prediction worth increased latency in decision-making? Are fast approaches sustainable under load? Are there task and structural differences between classifiers? This becomes especially important for staged Intrusion Detection System (IDS) implementations that separate initial screening from conclusive detection based on escalation probabilities. A holistic analysis of accuracy, error rates, latencies, throughput, and deployment scenarios is necessary. This paper uses an experimental cyber range containing seven virtual machine clusters for handling International Electrotechnical Commission (IEC) 61850, Modbus Transmission Control Protocol (TCP), and Distributed Network Protocol version 3 (DNP3) protocols [14, 15]. Layer 1 uses Random Forest (RF), K-Nearest Neighbors (KNN), and Extreme Gradient Boosting (XGBoost); Layer 2 uses Gated Recurrent Unit (GRU), and Deep Belief Network (DBN); and Layer 3 uses CNN, RNN, DNN, and Long Short-Term Memory (LSTM). The operational setting makes this trade-off unusually demanding. Substation and control-centre traffic combines periodic measurements, event-driven protection messages, supervisory commands, and maintenance activity. IEC 61850 GOOSE traffic, for example, is designed for rapid event delivery, whereas DNP3 and Modbus TCP are used for different polling and control patterns. A detector that treats these streams as interchangeable may learn protocol identity or traffic volume instead of attack behaviour. Protocol-aware cyber ranges and false-data-injection studies therefore provide a stronger basis for evaluation than generic enterprise-network datasets, although they still reproduce only part of the variation found in deployed grids [16–18].

Evaluation design is equally important. Packets from the same capture, attack execution, or operating state are correlated. If related records occur in both training and test sets, a model can appear to generalize while recognizing repeated conditions. This concern has long been raised in network intrusion-detection research and is especially relevant to cyber-physical datasets whose samples are produced by a small number of scripted scenarios [19–21]. Recent smart-grid benchmarks respond by publishing richer attack-specific datasets and process context, but their reported performance still varies with the split strategy, feature set, and degree of realism [22–24]. Accuracy values from different studies should therefore be compared only after these design choices are made explicit.

Latency also needs a precise measurement boundary. Model inference may occupy only a small fraction of the time between packet arrival and an actionable decision. Capture, parsing, feature construction, temporal-window assembly, memory transfer, queueing, thresholding, and alert delivery can add delay and variability. Batch size and accelerator use can improve average throughput while increasing the waiting time experienced by an individual event. Mean model time alone consequently cannot establish real-time suitability; the relevant evidence includes warm-up policy, hardware, software versions, concurrency,

repeated measurements, percentile latency, and the complete path through the cascade.

The need for a layer-wise evaluation follows from inherent trade-offs in the detection pipeline. A screening component deals with full traffic volume at the front end, requiring constraints on inference speed and the volume of retained traffic. A second model examines ambiguous cases. The final classifier acts upon a reduced but challenging workload. Each layer's merit depends on the job, the character of errors it commits, and its connection via routing probabilities to other layers. The contribution here lies in taking model accuracy scores as input to a constrained secondary analysis. It considers the validity of pairwise comparison under those constraints without raw predictions or routing logs.

This paper provides an auditable comparison based on the figures of the nine reported models. Inconsistencies in metrics and timescales are pointed out, along with latency calculations made according to each model's stated assumptions regarding routing and overhead costs. The study makes an additional contribution to methods: it determines which conclusions stand up to incomplete reports and demonstrates how to validate a new comparison in future. This analysis is neither concerned with finding an optimal deployment strategy nor with evaluating performance relative to real-world deployments. Sections 2–6 describe positioning, methodology, source extraction, calculation, and validation respectively.

2. Related Works

2.1. Intrusion Detection in Cyber-Physical Systems

Industrial control intrusion detection needs to comply with safety standards and bounded latencies. Surveys note heterogeneity among protocols and devices, scarcity of computational power, and the need to combine networking evidence with knowledge of process states [1, 3–5]. While runtimes reported by other work usually omit pre-processing, queuing, and windows, smart grid testbeds indicate the potential of protocol-specific features [16–18].

Smart grid intrusion detection literature covers signature scanners, specification and state estimation techniques, anomalies detected statistically, and supervised or self-supervised machine learning. Signatures provide interpretable matches for known malicious actions, but cannot recognise novel manipulations. Specification or estimation compares cyber activities against the anticipated states of associated processes and finds legitimate commands inconsistent with physical reality. Networked and physical relationships in machine-learning detectors capture non-linear dependencies between variables, subject to sufficient labels, stable operation, and robustness against drift or attack [1, 3–5].

Contextual considerations are demonstrated by protocol-specific research. The use of IEC 61850 messaging headers, patterns, timing, and GOOSE actions have been used to discriminate attacks from authentic substation events [16]. Protocol flaws and vulnerabilities revealed in the experimental misuse of DNP3 are unlikely to occur in non-specialist network datasets [17]. Cyber range exercises for injecting false data relate network activity to updated substations [18]. This research increases ecological validity, but comparisons suffer from differences in scripting attacks, minority-majority balance, feature extraction, and delay constraints.

2.2. Computational Profiles of Candidate Models

Random forests' predictions are unaffected by the number of training samples. RF iterates over a fixed forest of decision trees. While RF can learn compact and effective tree sets, XGBoost's boosted trees are evaluated equivalently [6]. Without indexing or approximation, the latency and resource consumption of kNN grow with the size of the reference set. This dependence makes kNN especially vulnerable to variations in training-set scale. Recurrent neural networks perform simple recurrent updates to ordinary sequences. Gated recurrent units and long short-term memory cells operate over sequences using gated hidden layers [10, 11].

DNNs execute dense linear algebra over inputs. DBN trains layers in a hierarchy of latent representations, usually starting with pretraining of stacked RBMs [9, 12]. CNN learns hierarchies of representations via shared convolutional filters and parallel computation [9, 13]. All complexities are ranked nominally unless otherwise modified by accelerations, sequence length, or mini-batching. Their computational profiles are summarised in Table 1.

Table 1. Model Families and Expected Inference Characteristics

Model family	Models	Inference driver	Expected scaling concern
Tree ensemble	RF, XGBoost	Tree depth and number	Model size and cache behaviour
Instance based	KNN	Distance to stored samples	Reference-set size and dimension
Recurrent	RNN, GRU, LSTM	Sequence length and hidden width	Serial dependency and window assembly
Feed-forward	DNN, DBN	Layer widths and depth	Matrix operations and memory
Convolutional	CNN	Kernel width, channels, length	Input arrangement and accelerator use

Note: Actual latency depends on implementation, hardware, batching, and preprocessing.

The classical models compared exhibit distinct deployment costs. Tree ensembles detect non-linear interactions among features, and the paths traversed by predictions are always finite if the forest or boosting iteration count is fixed. Introduction. Memory consumption and caching depend on the number and depth of trees. KNN retains the reference dataset, calculates distances during testing, and might suffer scalability problems unless query optimisations or approximations come into play. Those trade-offs introduce another dimension in terms of precision-speed tradeoffs worth considering along with benchmark measurements. Neural network architectures distribute memory in another fashion altogether. Linear neural networks benefit from matrix products and can solve specially engineered tabular problems. CNNs exploit parameter sharing across image regions, while RNNs capture sequence dynamics. The mechanisms for doing so involve GRU and LSTM gate functions for retaining information across various locations in a sequence. Deep Belief Networks establish a layered architecture by chaining probability distributions. That definition falls short of specifying any computational expense, however. Input dimensionality, hidden-layer widths, arithmetic precision, convolution kernels, computing hardware, compilation techniques, batching, and I/O overhead could shift the order of magnitude in several orders of magnitude. Comparative assessment demands a specific task definition for benchmarking.

2.3. Multi-objective Evaluation Gap

IDS metrics include accuracy, latency, false positives, false negatives, throughput, memory consumption, calibration, and extreme case performance. Condensing those seven characteristics into a composite measure, even with unknown weighting factors, masks operational relevance. Pareto front analysis preserves metrics as separate axes. Normalization and subsequent sensitivity analysis permit scalar ordering should that prove necessary [25, 26]. Layered approaches need per-stage adaptations either way. Screening handles larger data volumes with fewer consequences than classification downstream.

2.4. Benchmark Evidence and Evaluation Validity

Benchmarking research enables comparison between multiple datasets and model methods. However, comparing benchmark scores becomes impossible when there are discrepancies in the class distributions, train/test partitions, hardware platforms, and allowable delays. This audit will consider only one benchmark suite at a time. The goal is to audit that benchmark before presenting newer benchmarks whose results it fails to capture. ERENO audited an adversarial extension of IEC 61850 protocols. The work released attack-labelled datasets and showed substantial F1 variations caused by richer substation attributes [22]. In 2024, the IEC 60870-5-104 SCADA benchmark evaluated six architectural variants using a hybrid

virtual/physical testing ground, achieving drastically higher peak accuracies than those shown here [23]. Those benchmarks employed different tasks and datasets. The impact of split sensitivity has since been demonstrated for process-aware smart grids [24].

An additional limitation involves aggregate metrics without per-class results. Overall accuracy may reflect benign records or prevalent attacks rather than rare manipulation events. F1 macro average assigns equal importance to every class, whereas weighted F1 and micro average respond to distinct questions. False alarm and false detection probabilities are likewise conditional upon the selection of positive records and the threshold. Any reproduction therefore demands per-class confusion matrices, class-enablement, F1 type, calibration, uncertainties, and ideally a cross-subgroup analysis on independent captures or scenarios instead of single packets [19–21]. A multi-objective approach clarifies the trade-off among competing desiderata. Pareto optimisation highlights any algorithm whose published counterparts all achieve lower accuracy and higher speed, while a defined utility function discloses the trade-off parameters used to evaluate one option versus another [25, 26]. Within a cascaded architecture, however, each layer bears distinct responsibilities. Global dominance becomes a descriptive relationship rather than a selection principle, and overall evaluation should account for layer-wise misclassification costs, transmission probability, workload, computational requirements, and tail latency. This study contributes a selective interpretation of one multilayer test along with the necessary measurements for future deployment-level experiments.

3. Methods

3.1. Scope and Evidence Protocol

The quantitative evidence corpus consists of nine model rows in the performance table with the cyber-range description [14, 15]. This paper transcribed the model, layer, accuracy, precision, recall, F1, false alarm rate, missed alarm rate, and latency figure from the publication without reconstruction using any reported value. Verification applies when there exists a numerator, denominator, class definition, averaging method, and record to verify. Absent numerator-denominator pairs remain reported claims without adjustment for contradictory findings based on favourable choices between them. The study avoided meta-analysis across layers and other papers due to inconsistent labelling, task definitions, preprocessing methods, splits, and hardware. Recent papers mentioned in Section 2 provide context rather than serving as systematic search output. Systematic reviews would preregister the question and criteria in future iterations. Systematic searching would query IEEE Xplore, Scopus, Web of Science, ACM Digital Library, and Google Scholar using the dated combination “smart grid OR substation AND intrusion detection AND benchmarking AND latency AND machine learning”. Publication would include the deduplicated abstracts, two screeners’ decisions with rationales for excluding non-relevant documents, structured extraction forms, and bias assessments. Those steps are not claimed to have been executed here.

3.2. Multilayer Dataset and Evaluation Context

In IEC 61850, Modbus TCP, and DNP3 scenarios, the cyber range produced both benign and attack traffic. The following attributes were extracted: packet, transaction, timing, direction, size, port, and rate. Layer 1 swiftly eliminated clear cases; Layer 3 carried out more thorough classification, while Layer 2 confirmed ambiguous traffic [14]. This escalation logic is shown in Fig. 1.

The fact that latency values are not operationally interchangeable is made clear in Fig. 1. While deeper layers should only receive escalated observations, Layer 1 can handle the majority of traffic. Thus, routing probability and each model's latency affect the overall response time.

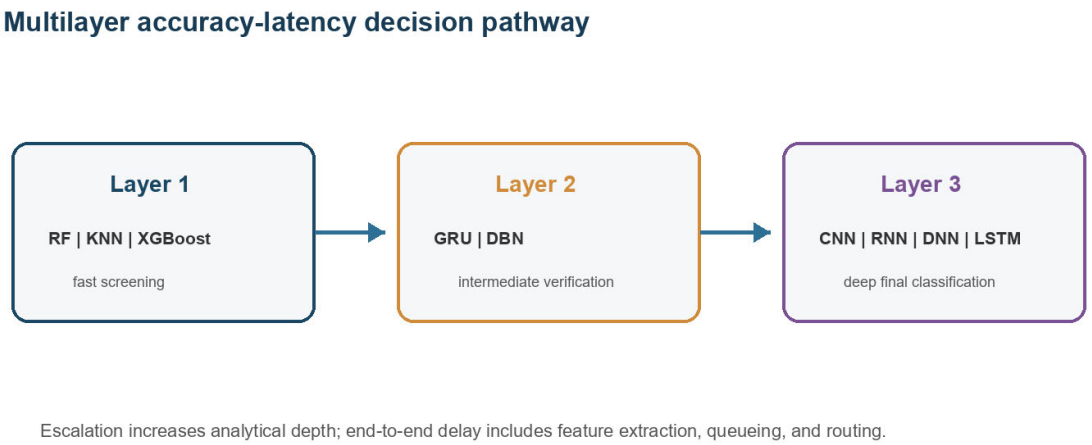


Fig. 1. Three-layer IDS pathway and the models included in the cross-layer analysis.

3.3. Accuracy, Latency, and Throughput

Eq. (1) defines accuracy, with TP, TN, FP, and FN standing for true-positive, true-negative, false-positive, and false-negative counts. The total diagonal of the confusion matrix can be used to apply the definition in a multiclass system, but per-class averaging for precision, recall, and F1 needs to be specified.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{1}$$

Eq. (2) provides the idealized single-stream throughput Q in observations per second for latency tau in milliseconds per observation. Since preprocessing and concurrency are not included in this reciprocal, it is an upper-bound interpretation rather than a measured throughput.

$$Q = \frac{1000}{\tau} \tag{2}$$

3.4. Pareto Dominance

If inequality Eq. (3) holds with at least one strict comparison, model a outranks model b on accuracy maximization and latency minimization. The Pareto front contains non-dominated models. Zero-rounded latencies cannot satisfy strict criteria.

$$A_a \geq A_b \wedge \tau_a \leq \tau_b \tag{3}$$

3.5. Normalized Utility and Cascade Latency (Model-only)

Eq. (4) computes a normalised penalty for loss utility. Parameter w represents weight assigned to the importance of accuracy error. Normalised latency e_τ and accuracy error e_A reflect observed scales. A smaller U indicates superior performance. This equation does not define any operational cost function universally. Instead, it acts as a sensitivity tool.

$$U_m(w) = we_{A,m} + (1-w)e_{\tau,m} \tag{4}$$

For a three-layer cascade, expected model-only latency is given by Eq. (5). Tau_1 is always incurred; p_{12} is

the probability of escalation from Layer 1 to Layer 2; and p_{23} is the conditional probability of further escalation. The equation shows why routing rates are essential to system-level claims.

$$E(T) = \tau_1 + p_{12}\tau_2 + p_{12}p_{23}\tau_3 \tag{5}$$

3.6. Alarm-cost Analysis

Expected classification cost in Eq. (6) combines FAR and MAR with false-alarm and missed-alarm costs c_{FA} and c_{MA}. Class prevalence and event rate should also be incorporated in deployment; the simplified form is used only to illustrate why equal accuracy can mask very different operational consequences.

$$C = c_{FA} \cdot FAR + c_{MA} \cdot MAR \tag{6}$$

4. Results

4.1. Unified Reported Results

Table 2 shows uniformly high reported accuracy but a latency span of more than 84 times among positive-latency models. RF is reported perfect with zero-displayed latency. DBN also has zero-displayed latency, but only 99.0% accuracy. XGBoost combines 99.9% accuracy with 0.000671 ms latency. DNN offers the strongest Layer 3 metrics at 0.001156 ms. KNN is the clear latency outlier at 0.056879 ms.

Table 2. Reported Predictive Performance and Latency across Layers

Layer	Model	Accuracy (%)	Precision (%)	Recall (%)	F1 (%)	FAR	MAR	Latency (ms)
1	RF	100.00	100.00	100.00	100.00	0.000000	0.0000	0.000000
1	KNN	99.90	99.90	99.90	99.90	0.000100	0.0005	0.056879
1	XGBoost	99.90	99.90	99.90	99.90	0.000004	0.0003	0.000671
2	GRU	99.00	99.00	99.70	93.30	0.029900	0.0034	0.001342
2	DBN	99.00	99.70	98.70	99.20	0.009100	0.0130	0.000000
3	CNN	98.35	98.79	99.16	98.91	0.040700	0.0083	0.001185
3	RNN	98.32	98.62	99.15	98.88	0.042200	0.0085	0.001133
3	DNN	99.49	99.53	99.79	99.66	0.014200	0.0021	0.001156
3	LSTM	98.58	98.66	99.47	99.06	0.041100	0.0053	0.001266

4.2. Accuracy-latency Frontier

Across all positive-latency rows, XGBoost dominates DNN because its reported accuracy is higher and its displayed latency is lower. DNN remains a distinct Layer 3 candidate, but cross-layer task differences prevent treating this global comparison as a replacement decision. XGBoost also dominates KNN, GRU, CNN, RNN, and LSTM on the two displayed numbers. The frontier is descriptive and provisional because hardware context, variance, and raw predictive outputs are unavailable. The Fig. 2 shows the reported accuracy-latency scatter for positive-latency models. The points on the plot do not establish a cross-layer deployment ranking.

The central trade-offs are quantified in Table 3. KNN is 84.77 times slower than XGBoost and has no reported accuracy advantage. DNN gains 1.17 accuracy points while being 2.03% slower than RNN. DNN is the strongest reported Layer 3 option since it is faster and more accurate than CNN and LSTM.

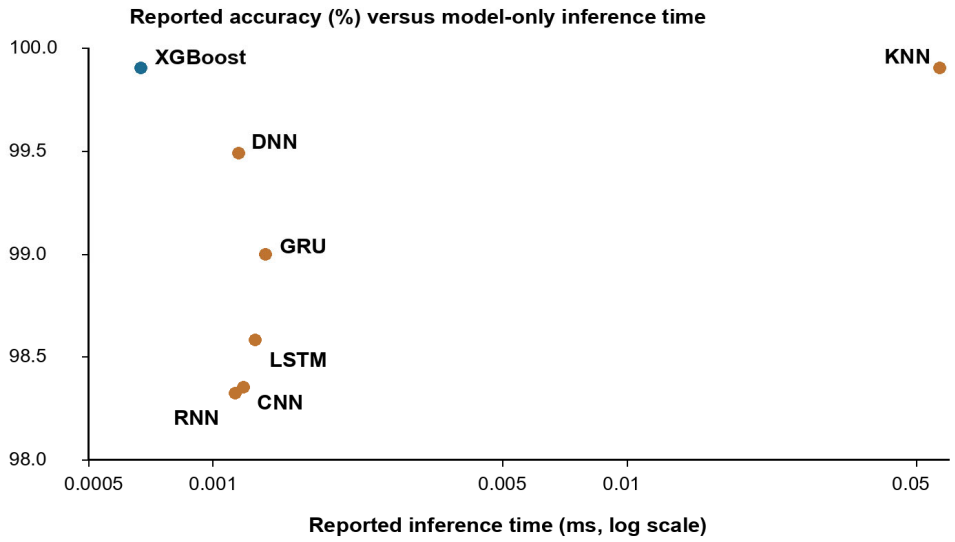


Fig. 2. Reported accuracy-latency scatter for positive-latency models. These points do not establish a cross-layer deployment ranking.

Table 3. Pairwise Latency and Accuracy Comparisons

Comparison	Latency ratio	Accuracy difference	Interpretation
KNN vs XGBoost	84.77 × slower	+0.00 pp	KNN is strictly dominated
DNN vs RNN	1.020×	+1.17 pp	2.03% latency premium for 1.17 pp
DNN vs CNN	0.976×	+1.14 pp	DNN is faster and more accurate
DNN vs LSTM	0.913×	+0.91 pp	DNN is faster and more accurate
GRU vs DNN	1.161×	−0.49 pp	DNN dominates reported pair

Note: Ratios and differences are derived from reported values. pp = percentage points.

4.3. Alarm-Rate Trade-Offs

Major false-alarm differences are concealed by aggregate accuracy, as Fig. 3 illustrates. RNN has the highest reported FAR at 0.0422, while Layer 1 rates are incredibly low. Layer 3 FAR is reduced by DNN to 0.0142, which is 65%–66% lower than CNN, RNN, and LSTM. When operator or downstream capacity is limited, DBN may be useful because it has a lower FAR than GRU.

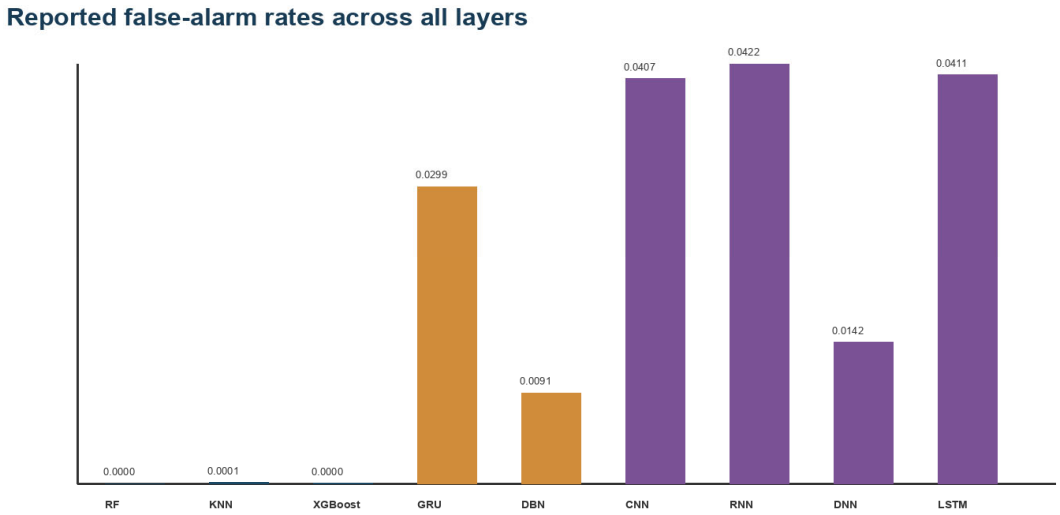


Fig. 3. Reported false-alarm rates by model and layer.

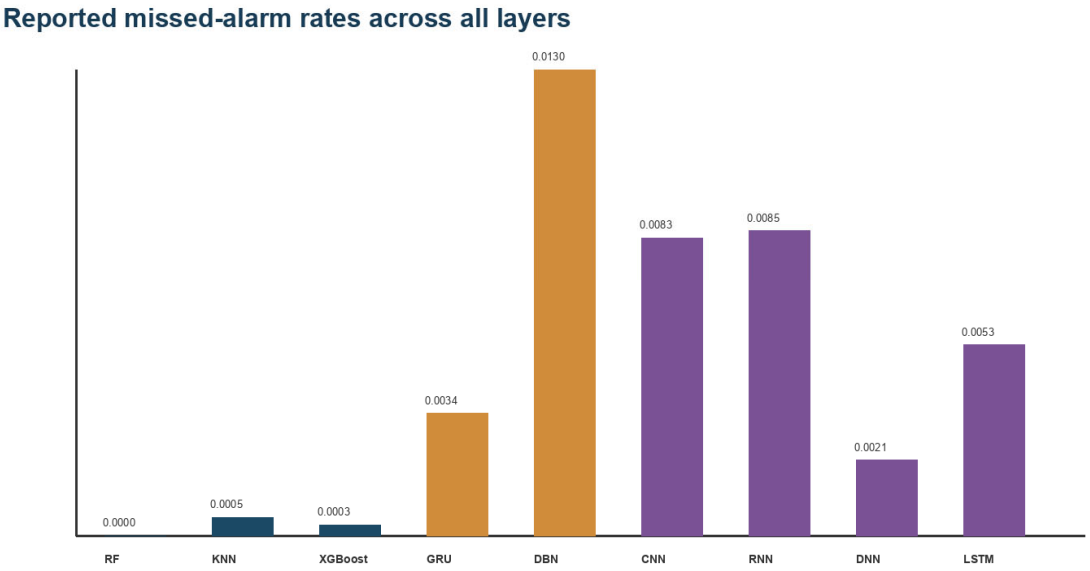


Fig. 4. Reported missed-alarm rates by model and layer.

The complementary missed-alarm view is shown in Fig. 4. Among deep models, DNN has the lowest reported MAR (0.0021), followed by GRU (0.0034) and DBN (0.0130). As a result, DBN misses more attacks at the reported operating point but manages false alarms more effectively than GRU. It is evident from Eq. (6) that the relative cost of these errors influences the choice of model.

4.4. Throughput and Utility Sensitivity

Latency is converted into an idealized but intuitive upper bound in Table 4. KNN corresponds to roughly 17,581 observations per second, whereas XGBoost corresponds to roughly 1.49 million. Between approximately 745,000 and 883,000 observations per second, the deep models cluster. Since reciprocal latency ignores concurrency and system overhead, these numbers cannot be presented as measured throughput.

Table 4. Idealized Reciprocal Throughput and Equal-Weight Utility

Model	Latency (ms)	Ideal Q (obs/s)	Speed rank	U(0.5)
KNN	0.056879	17,581	7	0.5298
XGBoost	0.000671	1,490,313	1	0.0357
GRU	0.001342	745,156	6	0.3094
CNN	0.001185	843,882	4	0.5015
RNN	0.001133	882,613	2	0.5100
DNN	0.001156	865,052	3	0.1619
LSTM	0.001266	789,889	5	0.4337

Note: Q is computed with Eq. (2) and excludes preprocessing, batching, queueing, and I/O. Lower U is better.

Among the displayed models-only numbers, XGBoost achieves the lowest positive latency and highest reported accuracy. Utility charts show its sensitivity to selection of normalization and weighting parameters, but the curves cannot prove that XGBoost should be adopted rather than others in a live situation. Zero-rounded RF and DBN predictions cannot be plotted accurately against each other, and KNN models are outperformed in Layer 1**.** The plot of the sensitivity of the normalized utility loss to the weight when placed on accuracy error is shown in Fig. 5. Table 5 presents the candidate roles for future validation.

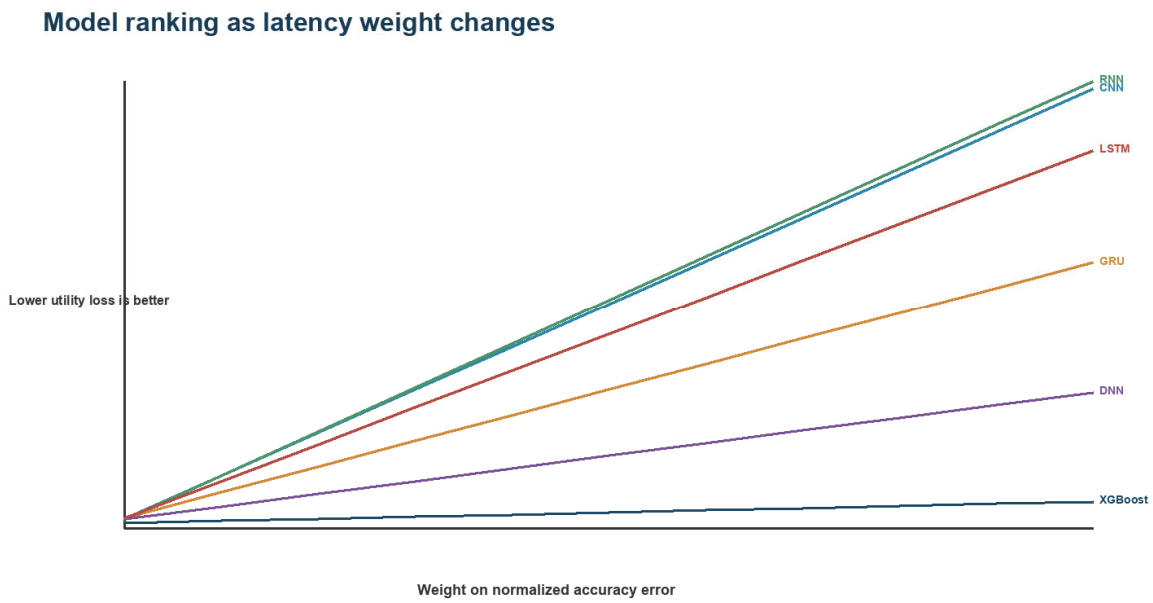


Fig. 5. Sensitivity of normalized utility loss to the weight placed on accuracy error.

Table 5. Candidate Roles for Future Validation

Layer	Model to test	Reported basis	Evidence still needed
1	XGBoost	99.9% accuracy; 0.000671 ms; very low FAR/MAR	Grouped split and hardware percentile latency
1 alternative	RF	Perfect reported metrics; zero-displayed latency	Timer resolution and confusion regeneration
2	GRU/DBN routing	GRU lower MAR; DBN lower FAR and zero-displayed latency	Calibrated thresholds and common metric definitions
3	DNN	Best Layer 3 metrics; 0.001156 ms	Per-class multiclass matrix and external validation
3 specialist	RNN/LSTM/CNN	Temporal or local inductive bias	Per-attack ablation showing specialist benefit

This analysis raises hypotheses, not suggestions, for controlled experimentation. Current summaries cannot confirm safety in a cascade or readiness for operational use.

4.5. Illustrative Cascade Latency Sensitivity (Model-only)

A $p12 = 0.05$, $p23 = 0.10$ configuration has 0.000744 ms expected sole-model execution. Configurations of 0.25, 0.50 give 0.001151 ms; 0.75, 0.90 give 0.002458 ms. After adding 0.1 ms, those configurations give 0.100744 ms, 0.101151 ms, and 0.102458 ms. For 1.0 ms additions, they give 1.000744 ms, 1.001151 ms, and 1.002458 ms. Those figures exclude any route-dependent preprocessing delays, queues, and tails. They illustrate sensitivities to an assumed cascade rather than empirical response times or capacities. No escalation probabilities, arrival rates, or latency distributions exist for calibration purposes. DBN requires similar sensitivity analyses, but it cannot interpret zero latency as a measurement. It needs a resolution bound and multiple samples. In order to measure $p12$ and $p23$, the experiment would need to release its direction assignments along with its results above threshold. End-to-end decision time at median, 95th percentile, 99th percentile, and worst case under typical workload, bursts, and attacks should be reported alongside queue depth and drops. DBN analysis does not admit a latency measurement of zero. Latency estimates require at least timer resolution and replication. Estimates of $p12$ and $p23$ need publication of the experiment's routing log. The log must be tagged by decision-latency observation and threshold. Replay

needs to include nominal loads, bursts, and attacks, and must report latency percentiles of median, 95th, 99th, and maximum end-to-end decision time, as well as queue level and event loss.

4.6. Reporting-Consistency Audit

As reported in Table 6, there is no contradiction in the existing trace logs that permits validation of the consistency between latency measurements and summary predictions for those same observations. Until consistent regeneration takes care of these inconsistencies, no confidence exists in any rank order among metrics.

Table 6. Measurement and Reporting Risks

Risk	Observed evidence	Effect on trade-off analysis	Remedy
Zero-rounded latency	RF and DBN shown as 0.0000 ms	Cannot order or compute throughput	Higher-resolution repeated benchmark
Narrative label swap	Large counts called FN	Confusion-derived metrics conflict	Generate text from raw confusion object
GRU F1 mismatch	P = 99.0%, R = 99.7%, F1 = 93.3%	Utility based on F1 is unreliable	Recompute with stated averaging
Missing benchmark context	No hardware/batch/percentiles	Latency portability is unknown	Publish environment and distributions
Cross-layer task differences	Models operate at different stages	Naive global ranking can mislead	Evaluate cascade end to end

Note: The audit separates measurement uncertainty from model performance.

5. Discussion

5.1. Interpreting the Apparent Winners

Among the positive Layer 1 values, XGBoost has the best reported accuracy-latency pair; KNN has the same reported accuracy and 84.77 times its displayed latency. This is a within-source observation, not evidence that XGBoost will be the best first-line detector on another device or traffic mix. RF's perfect reported metrics and zero-rounded time cannot be independently checked. Within Layer 3, DNN has the strongest reported summary metrics, but no per-attack or uncertainty analysis is available. Any explanation based on engineered features remains a hypothesis.

Layer 2 cannot be chosen from aggregate accuracy and latency alone. DBN and GRU both report 99.0% accuracy, while the reported FAR and MAR favor different error types. Their class definitions and confusion counts must first be reconciled. Threshold-routed combinations remain untested designs.

5.2. End-to-End LATENCY is the Real Deployment Metric

Model invocation is only one part of decision time. Packet capture, parsing, feature extraction, temporal-window assembly, normalization, queueing, interprocess transfer, logging, and response delivery are unmeasured here. Eq. (5) therefore yields only an illustrative model-only calculation. The benchmark boundary for any deployment study should begin when an observation is available to the IDS and end when its decision reaches the consuming security function.

The significance of escalation rates is further demonstrated by Eq. (5). Deeper layers may become overloaded by a highly accurate Layer 1 model that forwards too many uncertain events. An aggressive screen, on the other hand, might be quick but increase the number of missed attacks. In addition to queue length and tail latency, reports should include p_{12} and p_{23} under typical traffic, attack bursts, and adversarial flooding.

5.3. Tail Behaviour, Load, and Hardware

Rare delays that are important for protection can be hidden by mean latency. Following a warm-up, the

median, 95th, 99th, and maximum latency should be measured over several runs. Single observations, realistic micro-batches, CPU-only substation hardware, accelerators when available, and concurrent protocol streams should all be tested. Since KNN's inference cost depends on the data, it should be assessed as the stored reference set expands.

5.4. Raw-Prediction Metric Reconstruction

Per-class precision, recall, F1, and confidence intervals cannot be calculated from the aggregate table. The study provides, for each held-out observation, its true class, predicted class, class probabilities or scores, capture or attack-run group, model and fold identifiers, and the class-order mapping. The analysis should generate a confusion matrix once per model and fold, then derive every table and narrative count from that object. Precision, recall, and F1 must be reported for each class and as macro, micro, and weighted summaries with the chosen zero-division rule. The GRU discrepancy must be addressed from those outputs rather than by replacing 93.3% with the binary harmonic mean: its reported values may use different averaging rules.

Confidence intervals should resample independent capture or attack-run groups, repeating the complete metric calculation for each bootstrap draw; a binomial interval on individual packets would understate uncertainty when adjacent records are correlated. The number of groups, folds, bootstrap draws, seed, and interval method must be disclosed. Because these inputs are unavailable, this revision leaves all per-class results and intervals unreported rather than inventing them.

5.5. Accuracy is not enough

By giving preference to common classes, an IDS can attain high accuracy. For stealthy reconnaissance, command manipulation, and false-data injection in particular, per-attack precision, recall, F1, and confidence intervals are crucial. When attacks are infrequent, precision-recall curves provide more information than ROC curves. Whether confidence can safely control escalation is determined by calibration. Prevalence affects operational cost as well. While a small MAR might still be insufficient for high-impact attacks, a small FAR applied to millions of benign events can generate a significant alert volume. Class priors, event rates, response costs, and safety impact should all be added to Eq. (6). Instead of using a single default cutoff, a model should be selected at a defensible operating threshold.

5.6. Reproducibility and validity

Although it limits field generalization, the dataset was created in a controlled virtual range, which enhances repeatability. Correlated packets from the same capture or attack execution may be leaked by row-wise splitting. Evaluation must be capture-grouped, temporal, and leave-one-scenario-out [19, 20]. Vendor, firmware, load, and topology shifts should be tested by external hardware-in-the-loop and operational captures. Evaluation of any object with ground-truth labels, posterior probabilities, class order, split assignments, average strategy, and sample times should underlie tabulation and discussion for all metrics. All experimental parameters, including hardware configuration, software revisions, thread configurations, batch sizes, warming periods, replications, and percentiles, should be disclosed by the latency harness. An absolute zero measurement requires publication with an upper bound and sub-resolution precision.

5.7. Recommended Experimental Protocol

Any replication should adopt the same multiclass set of metrics, repeatable randomisation, calibration of posterior probabilities, nested parameter selection with equivalent computational budgets, and grouped cross-validation splits for each candidate. Accuracy, macro F1, false alarm rate, and missed alarm rate should complement model complexity, training cost, inferential power, efficiency, and latency tails. Packet-based replication implies independent sampling groups for any statistical inference. Then, under typical traffic and

attack floods, the entire cascade should be replayed. Routing probabilities, queue occupancy, dropped events, end-to-end decision time, and detection delay from attack onset should all be reported. The existing model-level evidence would be transformed into a deployable real-time claim by this protocol.

5.8. Security Implications

By creating borderline traffic that necessitates escalation and depletes deeper models, attackers can target the trade-off itself. It is necessary to implement rate limitation, admission control, backpressure, uncertainty caps, and fail-safe defaults. When confidence is low, a model shouldn't exchange latency for risky automatic actions. Poisoning and tampering risks are decreased by signed artifacts, provenance-aware retraining, drift monitoring, and protected feature pipelines.

6. Conclusion

This source-bounded secondary analysis summarizes nine reported smart-grid IDS models and identifies limits that affect every proposed architecture. XGBoost has a favorable displayed Layer 1 pair relative to KNN; DNN has the strongest reported Layer 3 aggregate metrics. These observations cannot be converted into a system recommendation. The GRU F1 inconsistency, confusion-matrix wording, zero-rounded latency values, missing hardware and repetition details, absent per-class predictions, and unknown escalation rates prevent reliable uncertainty estimates and end-to-end validation. The illustrative routing scenarios show that model-only latency changes with escalation, while unmeasured overhead may be much larger than the differences between models. A publishable operational claim requires raw-prediction regeneration of per-class precision, recall, F1, and group-aware confidence intervals; repeated, fully specified latency tests; and replay of the complete cascade under realistic routing and load. Until those data are available, the findings should be treated as provisional and should not guide deployment.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

J. E. Efiong conceptualized and initiated the study, including drafting the manuscript. A. Akinwale experimented and enhanced the initial draft of the manuscript. E. A. Olajubu supervised the experiment and reviewed the manuscript internally, G. A. Aderounmu provided the lab environment, funding and resources for the study. All authors had approved the final version.

Acknowledgment

This study received support in resources from the Cybersecurity Research Lab, OAU ICT-driven Knowledge Park, Obafemi Awolowo University, Nigeria.

References

[1] National Institute of Standards and Technology. (2023). *Guide to Operational Technology (OT) Security* (NIST Special Publication 800-82 Rev. 3). doi: 10.6028/NIST.SP.800-82r3

[2] Smart Grid Interoperability Panel–Smart Grid Cybersecurity Committee. (2014). *Guidelines for Smart Grid Cybersecurity* (NIST Interagency Report 7628 Rev. 1). doi: 10.6028/NIST.IR.7628r1

[3] Pliatsios, D., Sarigiannidis, P., Lagkas, T., & Sarigiannidis, A. G. (2020). A survey on SCADA systems: Secure protocols, incidents, threats and tactics. *IEEE Communications Surveys & Tutorials*, *22(3)*, 1942–1976. doi: 10.1109/COMST.2020.2987688

[4] Yaacoub, J.-P. A., Salman, O., Noura, H. N., Kaaniche, N., Chehab, A., & Malli, M. (2020). Cyber-physical

systems security: Limitations, issues and future trends. *Microprocessors and Microsystems*, *77*, 103201. doi: 10.1016/j.micpro.2020.103201

[5] Humayed, A., Lin, J., Li, F., & Luo, B. (2017). Cyber-physical systems security—A survey. *IEEE Internet of Things Journal*, *4(6)*, 1802–1831. doi: 10.1109/JIOT.2017.2703172.

[6] Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 785–794). ACM. doi: 10.1145/2939672.2939785

[7] Breiman, L. (2001). Random forests. *Machine Learning*, 45(1), 5–32. doi: 10.1023/A:1010933404324

[8] Cover, T. M., & Hart, P. E. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, *13(1),* 21–27. doi: 10.1109/TIT.1967.1053964

[9] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521(7553)*, 436–444. doi: 10.1038/nature14539

[10] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9(8)*, 1735–1780. doi: 10.1162/neco.1997.9.8.1735

[11] Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder–decoder for statistical machine translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1724–1734). Association for Computational Linguistics. doi: 10.3115/v1/D14-1179

[12] Hinton, G. E., Osindero, S., & Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7), 1527–1554. doi: 10.1162/neco.2006.18.7.1527

[13] Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press.

[14] Efiong, J. E. (2023). Development of a multilayered bio-inspired intrusion detection model for cyber-physical systems in the smart grid. Doctoral dissertation, Obafemi Awolowo University.

[15] Efiong, J. E., Akinyemi, B. O., Olajubu, E. A., Aderounmu, G. A., & Degila, J. (2023). CyberSCADA network security analysis model for intrusion detection systems in the smart grid. *Proceedings of Advances in Intelligent Systems, Computer Science and Digital Economics IV* (Vol. 158, pp. 481–499). Springer. doi: 10.1007/978-3-031-24475-9_41

[16] Ustun, T. S., Hussain, S. M. S., & Kalam, A. (2021). Machine learning-based intrusion detection for achieving cybersecurity in smart grids using IEC 61850 GOOSE messages. *Symmetry*, *13(5)*, 826. doi: 10.3390/sym13050826

[17] Darwish, I., Igbe, O., & Saadawi, T. (2015). Smart grid DNP3 vulnerability analysis and experimentation. *Proceedings of the 2015 IEEE 2nd International Conference on Cyber Security and Cloud Computing (CSCloud)* (pp. 141–147). IEEE. doi: 10.1109/CSCloud.2015.86

[18] Roomi, M. M., Mashima, D., & Chen, Y. (2020). False data injection cyber range of modernized substation system. *Proceedings of the 2020 IEEE International Conference on Communications, Control, and Computing Technologies for Smart Grids (SmartGridComm)* (pp. 1–7). IEEE. doi: 10.1109/SmartGridComm47815.2020.9302951

[19] Sommer, R., & Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. *Proceedings of the 2010 IEEE Symposium on Security and Privacy* (pp. 305–316). IEEE. doi: 10.1109/SP.2010.25

[20] Khraisat, A., Gondal, I., Vamplew, P., & Kamruzzaman, J. (2019). Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity*, *2,* 20, 1–22. doi: 10.1186/s42400-019-0038-7

[21] Ahmad, Z., Khan, A. S., Shiang, C. W., Abdullah, J., & Ahmad, F. (2021). Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, *32(1)*, e4150, 1–29. doi: 10.1002/ett.4150

[22] Quincozes, S. E., de Albuquerque, C. V. N., Passos, D. G., & Mosse, D. (2024). ERENO: A framework for generating realistic IEC 61850 intrusion detection datasets for smart grids. *IEEE Transactions on Dependable and Secure Computing*, *21(4),* 3851–3865. doi: 10.1109/TDSC.2023.3336857

[23] Arifin, M. A. S., Stiawan, D., Suprapto, B. Y., Susanto, Salim, T., Idris, M. Y., *et al*. (2024). A novel dataset for experimentation with intrusion detection systems in SCADA networks using IEC 60870-5-104 standard. *IEEE Access*, *12*, 170553–170569.

[24] Loughmari, M., Bouhadda, M., & El Affar, A. (2026). Towards reliable intrusion detection in smart grids: A multiclass, process-aware perspective. *Journal of Intelligent & Fuzzy Systems*.

[25] Deb, K. (2001). *Multi-objective Optimization Using Evolutionary Algorithms*. Wiley.

[26] Marler, R. T., & Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization*, *26(6),* 369–395. doi: 10.1007/s00158-003-0368-6

Copyright © 2026 by the authors. This is an open access article distributed under the Creative Commons Attribution License which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited (CC BY 4.0).