Advances in Networks
2026, Vol. 13, No. 2, pp. 40–53
https://doi.org/10.11648/j.net.20261302.11

Research Article

Stacked Deep Learning Ensemble Framework for IoT Network Intrusion Detection

Danjuma Israel Haruna[1], Boniface Kayode Alese[2], Gabriel Babatunde Iwasokun[3, *], David Bamidele Adewole[3], Ibraheem Temitope Jimoh[3], Victoria Ibiyemi Omoniyi[3], Mojisola Olajumoke Ogunseye[1]

[1]Department of Computer Science, Federal University of Technology, Akure, Nigeria
[2]Department of Cybersecurity, Federal University of Technology, Akure, Nigeria
[3]Department of Software Engineering, Federal University of Technology, Akure, Nigeria

Abstract

Network Intrusion Detection Systems (NIDS) are critical defense layers in modern IoT and IIoT environments, which are disproportionately targeted by evolving cyber threats. Classical machine learning approaches and individual deep learning models face documented limitations, including reliance on outdated datasets, susceptibility to class imbalance, and failure to leverage the complementary detection strengths of diverse architectures. This paper presents the formulation and evaluations of a Stacked Deep Learning Ensemble (SDLE) framework for binary network intrusion detection. The ensemble was based on Recurrent Neural Networks (RNN), Gated Recurrent Units (GRU), and Autoencoders as base learners with a Long Short-Term Memory (LSTM) network as the meta-learner. The framework is trained and evaluated on the ToN_IoT dataset, a contemporary IoT/IIoT benchmark comprising 461,043 records and nine attack categories, following a systematic three-stage preprocessing pipeline that incorporates label encoding, one-hot encoding, feature selection, SMOTE-based class balancing, and standard scaling. Experimental results demonstrate that the SDLE achieves 98.67% accuracy, 98.91% precision, 98.45% recall, and 98.68% F1-score, surpassing each base model and a simple voting ensemble by 0.78–1.94 percentage points. Preprocessing contributes a cumulative gain of 7.43 percentage points over raw data performance, with feature selection identified as the single most impactful step (+2.29%). The results also established that the platform provides an effective IDS model and empirical guidance for preprocessing and meta-learner design in deep learning ensemble systems.

Keywords

Network Intrusion Detection, Deep Learning Ensemble, Stacked Generalization, LSTM Meta-Learner, Autoencoder, ToN_IoT, SMOTE, IoT Security

*Correspondence: Gabriel Babatunde Iwasokun (gbiwasokun@futa.edu.ng)

Received: 30 June 2026; **Accepted:** 16 July 2026; **Published:** 22 August 2026

1. Introduction

The expansion of Internet of Things (IoT) and Industrial Internet of Things (IIoT) infrastructure across critical sectors, such as healthcare, energy, manufacturing, and public services, has introduced a new class of cybersecurity challenges. IoT devices are architecturally constrained in computational resources and memory, operate in uncontrolled physical environments, and communicate over heterogeneous protocols, making them disproportionately vulnerable to a broad spectrum of cyber-attacks [18]. Contemporary threats targeting these environments include Distributed Denial of Service (DDoS) flooding, ransomware, backdoor implantation, credential cracking, and Man-in-the-Middle (MITM) interception attack types that legacy signature-based Intrusion Detection Systems are fundamentally unable to detect without prior exposure [27].

Machine learning (ML) and deep learning (DL) methods have been widely applied to IDS research over the past two decades, demonstrating substantially stronger performance than rule-based approaches on standard benchmarks [20]. However, the dominant paradigm in this literature is training and evaluating individual models on static, often outdated datasets, which face well-documented limitations. First, the most widely used benchmarks, KDD'99 and NSL-KDD, were generated in 1998–1999 and do not reflect contemporary network traffic characteristics, encrypted communication, or IoT-specific attack patterns [13]. Second, individual DL architectures exhibit characteristic failure modes: RNNs and LSTMs model temporal dependencies but are sensitive to class imbalance; CNNs capture spatial patterns in traffic features but lack inherent sequential modelling; Autoencoders enable unsupervised anomaly scoring but do not produce direct class predictions [21].

Ensemble learning offers a principled solution to the limitations of individual models. By combining heterogeneous base learners whose errors are complementary, a stacked ensemble architecture can achieve lower bias and variance than any constituent model [11]. While ensemble methods are well-established in the classical ML IDS literature [3], the systematic stacking of architecturally diverse DL models, particularly combining supervised recurrent models with unsupervised reconstruction-based models under a learned meta-learner trained on modern IoT datasets, remains underexplored [14]. This paper proposes the Stacked Deep Learning Ensemble (SDLE) for network intrusion detection. The SDLE combines RNN, GRU, and Autoencoder base learners with an LSTM meta-learner, trained and evaluated on the ToN_IoT dataset under a systematic preprocessing pipeline. The research contributed to network intrusion detection by providing an end-to-end SDLE architecture for IoT IDS with a novel LSTM meta-learner, providing a systematic three-stage preprocessing ablation demonstrating the quantified contribution of each data preparation step, and an empirical demonstration of SDLE superiority over individual base learners and voting ensembles. The research also offers a comprehensive per-attack-type evaluation across nine IoT threat categories in addition to a computational efficiency analysis supporting real-world deployment considerations. The following sections present the related works, structured by thematic area, the methodology, including the dataset, preprocessing pipeline, and model architectures, the experimental results and discussion, and the conclusion drawn from the research.

2. Related Works

This section reviews prior work on ML- and DL-based IDS, structured across five themes: classical ML approaches, recurrent and convolutional DL models, autoencoder-based anomaly detection, ensemble and stacking methods, and dataset development. The review identifies six persistent gaps that motivate the proposed SDLE framework.

2.1. Classical Machine Learning Approaches

Supervised ML classifiers were among the earliest applied to intrusion detection and remain active in the literature. Akintoye et al. [2] evaluated six classifiers: Decision Tree, Gaussian Naive Bayes, K-Nearest Neighbour (KNN), Logistic Regression, Random Forest (RF), and Support Vector Machine (SVM) on UNSW-NB15 and NSL-KDD, with Decision Tree achieving 99.99% accuracy on UNSW-NB15. However, multi-class attack categorisation was not investigated, and evaluation on legacy datasets limits the translational value of reported metrics. Abdulkareem et al. [1] developed an ensemble classifier from Decision Tree, SVM, KNN, and Artificial Neural Network (ANN), attaining 99.8% efficiency on R2L attack detection but conceding poor performance on individual models, reinforcing the case for ensemble approaches. Wazirali [30] demonstrated KNN with hyperparameter tuning, achieving 98.49% accuracy, while identifying real-time inference latency as a critical unresolved deployment constraint. Sharif and Ahmed [25] combined SVM, KNN, RF, Neural Network, and an ensemble classifier with feature subset selection, attaining 98.19% precision; however, evaluation was confined to the NSL-KDD dataset, which does not reflect current attack patterns. A recurrent limitation across classical ML studies is that no single algorithm delivers uniformly strong performance across all attack classes, particularly on minority categories such as R2L and U2R attacks. This class imbalance vulnerability, compounded by the manual feature engineering burden of classical ML, provides the primary motivation for deep learning approaches.

2.2. Deep Learning for Network Intrusion Detection

CNN-based IDS approaches have demonstrated strong performance on structured traffic features. Waad et al. [29] achieved 99.8% accuracy with a CNN-Naive Bayes hybrid; however, evaluation on additional datasets and architectural simplification were not considered, raising generalisability concerns. More recent attention-based and transformer architectures have demonstrated improved detection of subtle, low-volume attacks by dynamically weighting the contribution of individual traffic features. Wu et al. [32] proposed RTIDS, a transformer-based intrusion detection model whose self-attention layers weight traffic features adaptively; the model attains strong accuracy but carries a parameter count several orders of magnitude larger than recurrent alternatives, which constrains deployment on resource-limited IoT gateways. Halbouni et al. [15] demonstrated that a CNN-LSTM hybrid on the CICIDS2017 dataset captures both spatial and temporal patterns, outperforming single-architecture models, though class imbalance was not explicitly addressed. LSTM networks have been consistently identified as strong performers for sequential traffic data. Isife et al. [17] conducted a rigorous comparison of DNN, LSTM, and GRU architectures on CICIDS2017 and CIDDS, with LSTM achieving 98.09% accuracy, 98.14% precision, and 99.69% True Negative Rate, the strongest result among the three. Critically, class imbalance in the skewed datasets was not addressed, potentially inflating the reported metrics on the majority classes. Almiani *et al.* [4] applied an RNN enhanced with backpropagation for fog-based IoT IDS, recording 94.27% detection rate and 92.18% accuracy on resource-constrained edge environments, but noted scope for further model enhancement. Ullah and Mahmoud [28] proposed an RNN anomaly detection model for IoT networks on the IoT-23 dataset, highlighting the sensitivity of RNN models to the vanishing gradient problem for long traffic sequences and identifying GRU and LSTM as more stable alternatives.

2.3. Autoencoder-Based Anomaly Detection

Autoencoders have gained sustained interest in IDS applications due to their ability to model normal traffic distributions without labelled attack data. Alsoufi *et al.* [7] designed an anomaly-based IDS using a Sparse Autoencoder (SAE) for feature compression followed by a CNN classifier, demonstrating effective dimensionality reduction and anomaly scoring for IoT networks. Ravi *et al.* [24] proposed a recurrent feature fusion ensemble meta-classifier using KPCA for feature selection, achieving strong results across UNSW-NB15 and CICIDS2017, though the study noted that transformer-based models may not align well with the statistical feature structure of IDS datasets. Hybrid frameworks combining LSTM Autoencoders with Random Forest classifiers have demonstrated 96.58% detection accuracy whilst maintaining false alarm rates below 2%, with blockchain integration for tamper-proof alert logging [16].

A consistent finding across autoencoder-based IDS studies is the complementarity of reconstruction-based anomaly scoring with supervised classification: the former excels at detecting novel, low-prevalence attack types that supervised models systematically miss, whilst the latter provides direct categorical classification. This complementarity provides the theoretical foundation for combining Autoencoders with RNN and GRU models within a stacked ensemble.

2.4. Ensemble and Stacking Approaches

Classical ML ensemble methods are well-established in IDS research. Ali *et al.* [3] evaluated a stacking ensemble using KNN, SVM, and Random Forest with XGBoost as the meta-learner, achieving 98.24% weighted F1-score on NSL-KDD; however, evaluation on a single dataset restricted assessment of generalizability. Alotaibi and Ilyas [5] applied Random Forest, KNN, Logistic Regression, and Decision Tree stacking on ToN_IoT, attaining 98.63% accuracy and 98.61% F1-score, but did not address multi-class classification. Olasehinde [23] developed a stacked ensemble of KNN, Naive Bayes, and Decision Tree with multiple meta-classifiers, achieving 99.01% accuracy, though the high execution times limited the viability of real-time deployment.

Deep learning ensemble integration remains the frontier of IDS research. Bingu and Jothilakshmi [8] combined LSTM, CNN, RNN, GRU, and DNN in an ensemble framework for cloud and SDN environments, achieving approximately 99.8% detection rate; Quality-of-Service requirements were not addressed. Farhan *et al.* [12] applied a Sequential DNN with Extra Tree Classifier feature selection on UNSW-NB15, reducing features from 43 to 8 whilst maintaining 97.93% accuracy, demonstrating that model compression and performance are not mutually exclusive. Gyimah et al. [14] surveyed ensemble learning for cybersecurity IDS, identifying the stacked DL ensemble paradigm as the highest-priority direction for future research, particularly when combining models with complementary temporal and spatial modelling strengths.

2.5. Dataset Development and Currency

The quality and contemporary relevance of evaluation datasets is a foundational determinant of IDS model validity. Tavallaee *et al.* [26] developed NSL-KDD as an improvement over KDD'99, eliminating redundant records and enabling more reliable benchmarking; however, the dataset reflects 1990s network traffic and attack patterns, predating cloud, IoT, and encrypted communication. Moustafa and Slay [22] developed UNSW-NB15 at the University of New South Wales, which provides nine attack categories and eliminates redundancy, representing a significant modernisation over legacy benchmarks. Alsaedi *et al.* [6] introduced ToN_IoT, generated

from a realistic IoT/IIoT testbed at the UNSW Canberra Cyber Range, encompassing 22.3 million records across nine contemporary attack categories, including ransomware, MITM, and XSS, making it the most comprehensive contemporary IoT IDS benchmark. Guerra et al. [13] critically examined the gap between academic IDS benchmarks and real-world network conditions, demonstrating that models trained on legacy datasets exhibit substantially degraded performance when evaluated on modern traffic distributions. This finding directly motivated the use of ToN_IoT as the primary evaluation benchmark in the research.

2.6. Synthesis and Identified Research Gaps

The literature review reveals six substantive gaps that the proposed SDLE framework addresses. First, the persistent reliance on KDD'99 and NSL-KDD in evaluation limits the ecological validity of reported results. Second, whilst individual DL architectures have been extensively studied, their systematic integration within principled stacked ensemble frameworks specifically combining supervised recurrent models with unsupervised reconstruction-based models remains underexplored. Third, the majority of reviewed studies frame IDS as a binary classification without investigating per-attack-type performance across minority threat categories. Fourth, class imbalance is inadequately addressed in a large proportion of reviewed studies. Fifth, systematic ablation of preprocessing pipelines, isolating the contribution of each step, is rarely conducted. Sixth, the use of LSTM as a meta-learner for stacked DL ensembles, whilst theoretically motivated, has not been empirically validated on modern IoT datasets. The proposed SDLE framework directly targets all six gaps.

3. Methodology

This section describes the SDLE framework, including the dataset, the preprocessing pipeline, the base learner architectures, the meta-learner, and the training settings. Figure 1 gives an end-to-end view of the SDLE architecture: ToN_IoT data passes through the preprocessing pipeline, then three base models (RNN, GRU, and Autoencoder) produce outputs that the LSTM meta-learner combines.

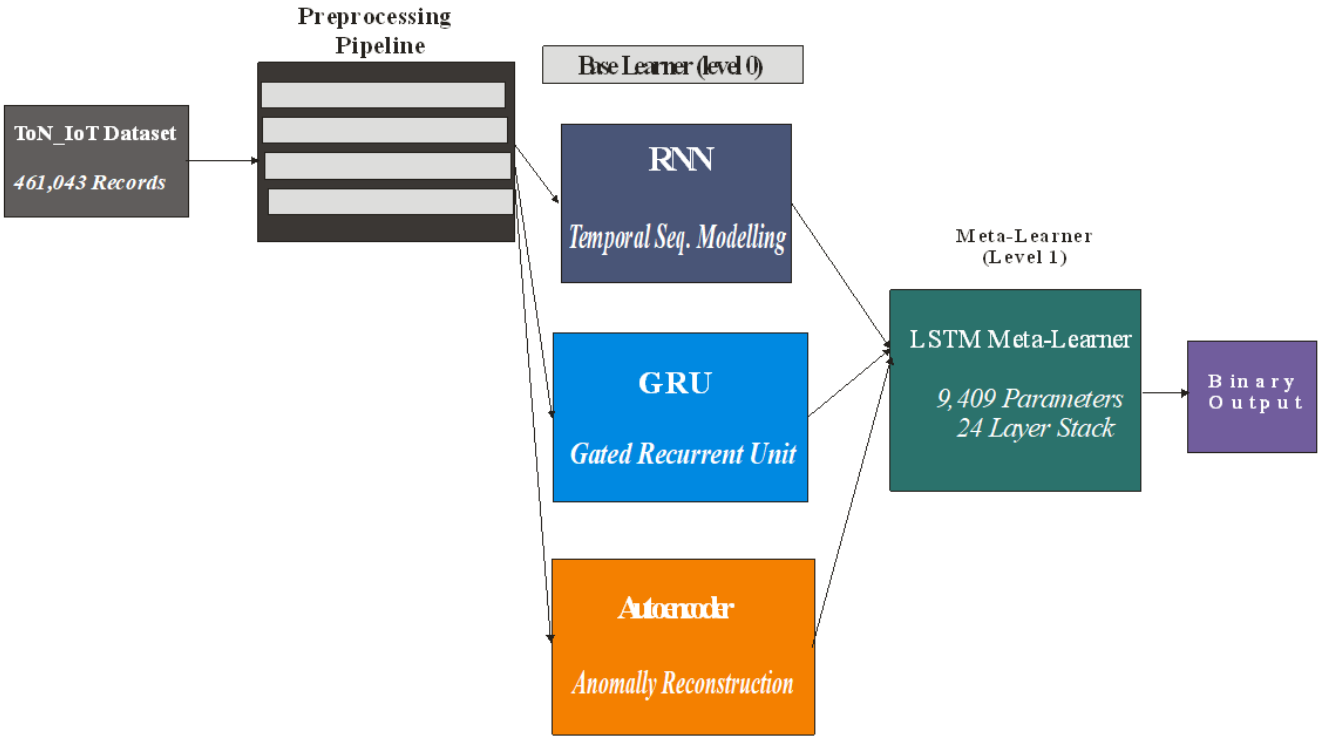

Figure 1. *Stacked Deep Learning Ensemble (SDLE) Framework Architecture.*

3.1. Base Learner Architectures

The SDLE uses three base learners with complementary strengths. Each one operates on the 45-dimensional ToN_IoT feature vector produced by the preprocessing pipeline. The RNN captures time dependencies across the sequence of network flow features through its hidden state recurrence, as shown in Equation (1). The output is derived based on Equation (2).

$$h_t = \sigma_h(W_{xh}x_t + W_{hh}h_{t-1} + b_h) \quad (1)$$

$$y_t = \sigma_y(W_{hy} \times h_t + b_y) \quad (2)$$

x_t is the 45-feature ToN_IoT input at step t, h_t is the hidden state, W_{xh} and W_{hh} are the input and recurrent weight matrices learned on the balanced ToN_IoT training set, b_h is the bias, and σ_h is the tanh activation. The output y_t is the attack probability for the flow.

The GRU extends the RNN with an update gate and a reset gate that reduces the vanishing gradient problem while using fewer parameters than an LSTM, which suits resource-constrained IoT deployment. The gate and state equations applied to the ToN_IoT feature vector are given in Equations (3) to (6):

$$z_t = \sigma(w_{xz}x_t + w_{hz}h_{t-1} + b_z) \quad (3)$$

$$r_t = \sigma(w_{xr}x_t + w_{hr}h_{t-1} + b_r) \quad (4)$$

$$\tilde{h}_t = tanh(w_{xh}x_t + U(r_t \odot \ h_{t-1}) + b_h) \quad (5)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \quad (6)$$

z_t represents the update gate, r_t represents the reset gate, and $\tilde{h}_t$ represents the candidate hidden state. w_{xz} and w_{hz} represents the update gate weights, w_{xr} and w_{hr} represent the reset gate weights, w_{xh} and w_{hh} represents the candidate hidden state weights. The update gate z_t determines how much of the previous hidden state h_{t-1} should be carried forward to the current hidden state h_t, while the reset gate r_t controls how much of the previous hidden state to forget. In these equations z_t controls how much of the previous state is carried forward, r_t controls how much is forgotten, $\tilde{h}_t$ is the candidate state, and $\odot$ is element-wise multiplication. The weights W, U, and biases b are learned on the same ToN_IoT training set.

The Autoencoder is trained on the 300,000 normal ToN_IoT instances only, so it learns the distribution of normal traffic. The encoder compresses the 45 input features into a 20-dimensional latent code through Equation (7), and the decoder reconstructs the 45 features through Equation (8):

$$z = \sigma_e(W_e x + b_e) \quad (7)$$

$$\hat{x} = \sigma_d(W_d z + b_d) \quad (8)$$

The reconstruction loss in Equation (9) is small for normal traffic and large for attack traffic, so it acts as an anomaly score for the meta-learner:

$$L(x, \hat{x}) = \|x - \hat{x}\|_2^2 = \sum_{i=1}^{n}(x_i - \hat{x}_i)^2 \quad (9)$$

x is the 45-feature input, z is the latent code, $\hat{x}$ is the reconstruction, W_e and W_d are the encoder and decoder weights, b_e and b_d are the biases, and σ_e and σ_d are the ReLU and linear activations. All base models use two stacked recurrent layers of 64 hidden units, the Adam optimiser with a learning rate of 0.001, dropout of 0.2, a batch size of 32, and early stopping with a patience of 15 epochs. Section 3.4 reports the full hyperparameter search and the settings required to reproduce these results.

3.2. LSTM Meta-Learner

The meta-learner takes the three base model outputs, the RNN and GRU attack probabilities, and the normalised Autoencoder reconstruction error, and forms the meta-feature vector in Equation (10):

$$\varphi(x_i) = [p^{RNN}(x_i), p^{GRU}(x_i), p^{AE}(x_i)]^T \in \mathbb{R}^3 \quad (10)$$

This 3-dimensional vector is reshaped as a short sequence of shape (None, 3, 1) and fed to two stacked LSTM layers of 32 units each, with tanh activation and dropout of 0.2, then a Dense layer of 16 units with ReLU activation, and a Dense output layer of 1 unit with sigmoid activation, for 9,409 trainable parameters. The LSTM gates let the meta-learner weight the base predictions by their reliability and model the interaction between them, which a simple vote or average cannot do. This is the main design contribution of the SDLE.

3.3. Training and Evaluation Protocol

All models are trained with a 70/15/15 train/validation/test split using stratified sampling to preserve attack class distributions. The evaluation employs accuracy, precision, recall, F1-score, and Receiver Operating Characteristic - Area Under the Curve (AUC-ROC) as primary metrics, complemented by per-attack-type analysis and a confusion matrix. Five-fold cross-validation is performed for statistical robustness, with mean and standard deviation reported across folds. The complete implementation is in Python using TensorFlow 2.12.0 on an NVIDIA GeForce RTX 3080 GPU (Ubuntu 22.04 LTS). Figure 2 illustrates the PRISMA-informed literature screening workflow applied in the review phase. 312 candidate papers retrieved, 147 screened, 68 included in the final synthesis.

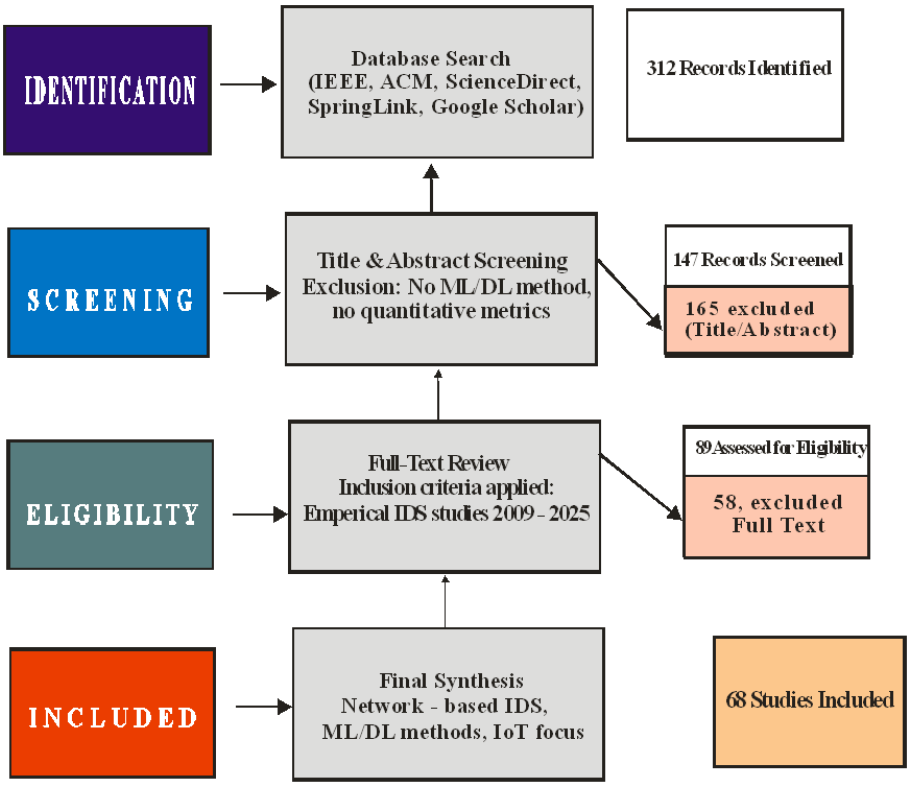

Figure 2. *Literature identification and screening flow.*

3.4. Implementation Details and Reproducibility

To support independent replication, the complete experimental configuration is reported here. All experiments were executed on a workstation with an Intel Core i7-10700K processor (8 cores, 16 threads, 3.80 GHz), 16 GB of DDR4-3600 memory, an NVIDIA GeForce RTX 3080 GPU (10 GB GDDR6X, CUDA compute capability 8.6), and a 512 GB NVMe solid-state drive, running Ubuntu 22.04 LTS with Linux kernel 5.15.0. The software stack comprised Python 3.8, TensorFlow 2.12.0 with GPU acceleration, Keras 2.12.0, NumPy 1.24.3, Pandas 1.5.3, Scikit-learn 1.2.2, Imbalanced-learn 0.10.1 for SMOTE [9], and Matplotlib 3.6.3 with Seaborn 0.12.1.

A single global random seed of 42 was set for the Python random module, NumPy, and TensorFlow before every run. The same seed governed the stratified train, validation, and test partition, the SMOTE neighbour sampling, and weight initialisation, so that each reported figure can be regenerated exactly. Weights were initialised with the Glorot uniform scheme.

Optimisation used Adam with a learning rate of 0.001, $\beta 1 = 0.9$, $\beta 2 = 0.999$, and $\varepsilon = 1 \times 10^{-7}$. The supervised base learners and the meta-learner were trained against binary cross-entropy loss, while the Autoencoder minimised mean squared reconstruction error. Training ran with a batch size of 32 for a maximum of 100 epochs. L2 regularisation of $\lambda = 0.0001$ and gradient-norm clipping at 5.0 were applied throughout, and the learning rate was reduced by a factor of 0.1 whenever validation loss failed to improve for five consecutive epochs. Training stopped early when validation loss showed no improvement for 15 consecutive epochs, and the weights from the best validation epoch were restored. Under these criteria the RNN converged at epoch 72, the GRU at 68, the Autoencoder at 65, and the LSTM meta-learner at 58, all well inside the 100-epoch ceiling.

The hyperparameters were determined experimentally rather than adopted from previous literature. A grid search was conducted on the validation partition over hidden units {32, 64, 128}, dropout {0.1, 0.2, 0.3}, learning rate {0.0001, 0.001, 0.01}, and batch size {16, 32, 64}, and the configuration that minimised validation loss was selected for the final models. This produced two stacked layers of 64 hidden units with dropout of 0.2 for both recurrent base learners, and an encoder path of 45-32-20 units for the Autoencoder. To prevent leakage, the SMOTE resampling and the standard-scaling parameters μ and σ were fitted on the training partition alone and then applied unchanged to the validation and test partitions, which were left at their original class distribution.

3.5. Rationale for the LSTM Meta-Learner

Stacked generalisation conventionally combines base learners with a simple meta-model, most often linear regression, logistic regression, or a tree ensemble [31]. The choice of an LSTM in place of these conventional meta-classifiers is motivated by the composition of the meta-feature vector rather than by architectural novelty for its own sake.

The three quantities entering the meta-learner are heterogeneous in kind. Two are calibrated attack probabilities from supervised recurrent classifiers, and the third is a normalised reconstruction error produced by an unsupervised model that never observed an attack during training. These signals are not interchangeable, they carry different scales, and their individual reliability varies with the traffic being examined. In high-volume flooding traffic the recurrent classifiers are confident and the reconstruction error adds little, whereas in sparse or previously unseen traffic the supervised probabilities drift towards the decision boundary and the reconstruction error becomes the more informative signal.

Conventional meta-classifiers cannot express this behaviour efficiently. XGBoost and Random Forest partition the meta-feature space with axis-aligned thresholds, so an input-conditional re-weighting of the three signals must be approximated by a large number of splits, which fragments the probability estimates and degrades calibration near the decision boundary. A multilayer perceptron applies a single weighting learned once for the whole distribution and has no mechanism for modulating one input on the evidence supplied by another. The LSTM, by contrast, processes the meta-feature vector as a short sequence, and its input, forget, and output gates apply multiplicative, input-dependent weighting, so the contribution of each base learner is gated according to the values of the others. This is precisely the selective trust the ensemble requires.

Empirical support for this reasoning is already present in the results. The simple voting ensemble, which aggregates the same three base learners under a fixed rule, reaches 97.68% accuracy, which is below the 97.89% of the strongest single base learner, whereas the LSTM meta-learner reaches 98.67%. A static combination rule therefore degrades the ensemble, while a learned gated combiner recovers and exceeds the best individual model. Because the base learners are identical in both cases, the 0.99-point gap between the two aggregation strategies isolates the contribution of the meta-learner itself.

This design also separates the SDLE from the deep ensembles reported previously. Bingu and Jothilakshmi [8] concatenate five supervised deep models under a static aggregation rule, and Ravi et al. [24] apply a meta-classifier to fused features rather than to model outputs. Neither combines supervised sequence models with an unsupervised reconstruction detector under a learned recurrent combiner, and neither validates such a design on a contemporary IoT benchmark. A direct empirical comparison against XGBoost, Random Forest, and multilayer perceptron meta-learners on the identical meta-feature set is the natural next step and is identified in Section 4.7 as planned work.

4. Experimental Study

The system was implemented using Python 3.8+ with libraries including Pandas, NumPy, Scikit-learn for data processing, and TensorFlow 2.x/Keras for deep learning models. The ToN_IoT (Telemetry of Network and Internet of Things) dataset, developed at the UNSW Canberra Cyber Range and IoT Labs by Alsaedi et al. [6], was used for the study and also served as the primary evaluation benchmark. The complete dataset comprises 22,339,021 records, while its curated research subset contains 461,043 records comprising 300,000 normal traffic instances and 161,043 attack instances distributed across nine attack categories (20,000 instances per category, except MITM with 1,043). The nine attack types DoS, DDoS, Ransomware, Backdoor, Injection, XSS, Password Cracking, MITM, and Scanning collectively represent the contemporary IoT threat landscape. The dataset includes 45 features following preprocessing, drawn from multi-source network traffic logs, operating system events, and IoT sensor telemetry.

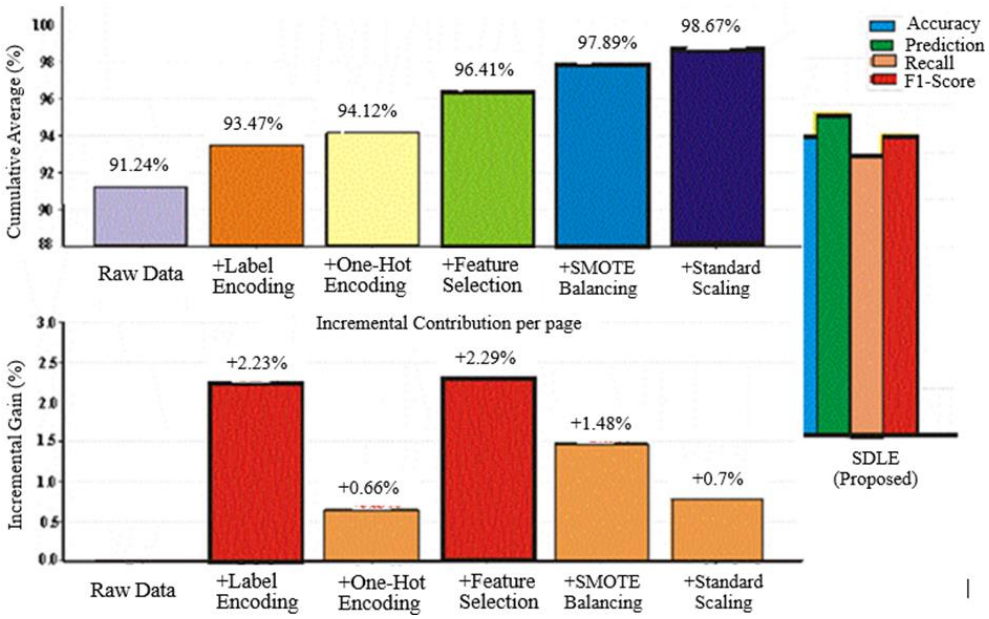

Figure 3. *Cumulative accuracy improvement across preprocessing stages.*

Three preprocessing configurations were used to measure the contribution of each step. Configuration 1 applies label encoding and one-hot encoding to the categorical features. Configuration 2 adds SMOTE class balancing to correct the imbalance, since normal traffic is about 65% of the research subset. Configuration 3 adds standard scaling, that is, z-score normalisation, as the last step. The full pipeline, which runs in sequential order of label encoding, feature selection that reduces 156 features to 45, one-hot encoding, SMOTE, and standard scaling, is the best configuration based on the ablation results in Section IV. Figure 3 shows the cumulative accuracy effect of each stage.

The experimental results were presented in order of increasing specificity. While the overall detection outcome is first established, the behaviour of the model on each attack category is then examined, the contribution of each preprocessing stage is isolated, and the findings are finally benchmarked against comparable studies. Across the full 461,043-sample test set, the SDLE correctly classified 454,607 of 461,043 instances, misclassifying 3,825 normal records as attacks and missing 2,611 genuine attacks. This outcome corresponds to 98.67% overall accuracy and an AUC-ROC of 0.9954.

4.1. Overall Model Performance

Table 1 presents the overall binary classification performance of all five evaluated models on the ToN_IoT test set (461,043 samples). The SDLE achieves the highest scores across all metrics, with 98.67% accuracy, 98.91% precision, 98.45% recall, 98.68% F1-score, and AUC-ROC of 0.9954. These results represent an improvement of 0.78 percentage points over the best individual base model (GRU) and 0.99 points over the simple voting ensemble.

Table 1. *Overall Binary Classification Performance on ToN_IoT Test Set (n = 461,043).*

Model	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)	AUC-ROC
RNN (Base)	97.46	97.82	97.24	97.53	0.9873
GRU (Base)	97.89	98.14	97.68	97.91	0.9911
Autoencoder (Base)	96.73	96.98	96.52	96.75	0.9814
Voting Ensemble	97.68	97.95	97.45	97.70	0.9891
SDLE (Proposed)	98.67	98.91	98.45	98.68	0.9954

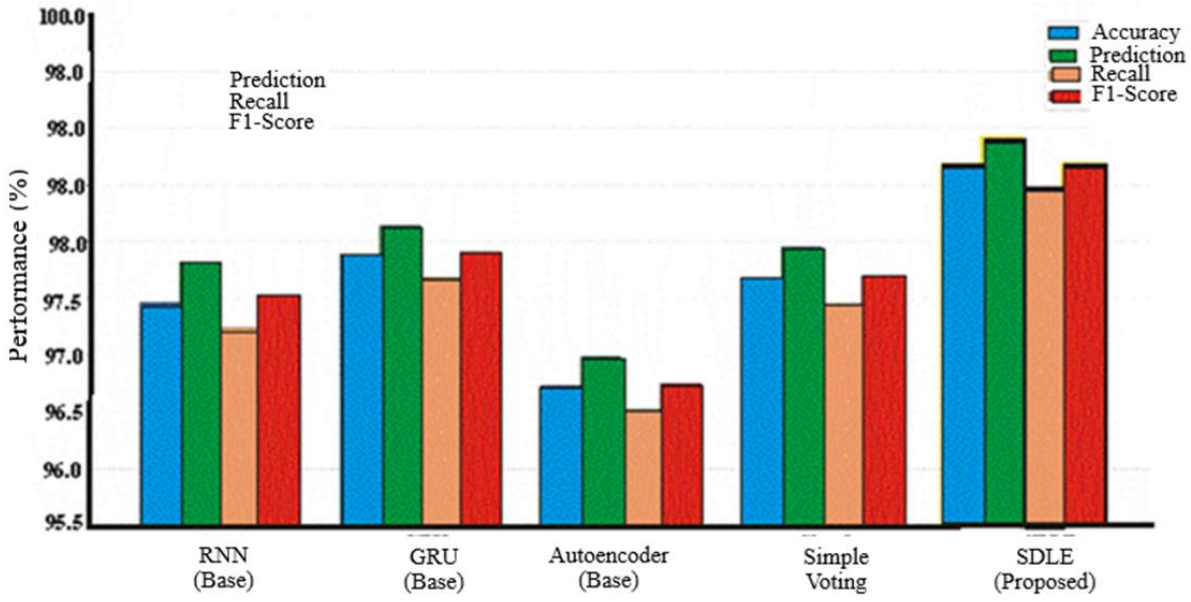

Figure 4. *Model Performance Comparison on ToN_IoT Dataset.*

The superiority of the SDLE over the simple voting ensemble (+0.99%) confirms that a learned, non-linear meta-combination strategy is essential to fully realize the potential of

DL ensemble architectures in intrusion detection. The GRU's strong individual performance (97.89%) relative to the RNN (97.46%) is consistent with prior literature attributing GRU's efficiency advantage to its simplified two-gate architecture. The Autoencoder's comparatively lower accuracy (96.73%) reflects its unsupervised learning paradigm, which is not optimised for binary decision boundaries; however, its complementary anomaly scoring mechanism contributes measurably to SDLE ensemble performance. Figure 4 provides a visual comparison of all four performance metrics across the five models. It shows a grouped bar chart comparing Accuracy, Precision, Recall, and F1-Score across RNN, GRU, Autoencoder, Voting Ensemble, and the SDLE. The SDLE (rightmost group) achieves the highest values across all four metrics.

4.2. Per-Attack-Type Performance

Table 2 presents per-attack-type F1-scores for the SDLE model. Detection performance is uniformly strong across all nine attack categories, with F1-scores ranging from 97.51% (MITM) to 99.14% (DDoS). The minimum detection rate of 97.51% across all attack types confirms robust generalisation across the full attack taxonomy.

Table 2. *Per-Attack-Type SDLE Performance on ToN_IoT.*

Attack Category	Samples	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
DoS	20,000	98.92	99.08	98.76	98.92
DDoS	20,000	99.14	99.31	98.97	99.14
Ransomware	20,000	98.45	98.67	98.23	98.45
Backdoor	20,000	98.67	99.01	98.34	98.67
XSS	20,000	97.89	98.21	97.57	97.89
Injection	20,000	98.34	98.56	98.12	98.34
Password Cracking	20,000	98.12	98.45	97.79	98.12
Scanning	20,000	98.56	98.87	98.25	98.56
MITM	1,043	97.51	97.81	97.21	97.51

Examining the specific attack categories (Table 2), detection was strongest for the volumetric classes, with DDoS reaching an F1-score of 99.14% and DoS 98.92%. This pattern is consistent with the temporal-modelling strengths reported by Almiani et al. [4] and Ullah and Mahmoud [28], whose recurrent architectures likewise detected high-volume flooding most reliably, since such attacks produce distinctive, repetitive packet sequences. The lowest score among the standard-sized categories was recorded for XSS (97.89%), in agreement with Isife et al. [17], who similarly identified application-layer attacks with variable encoding as the hardest class for network-flow-based detectors. The 97.51% F1-score for MITM is attributable to its limited sample size of 1,043 instances rather than to a modelling deficiency, a constraint that reinforces the dataset-currency concerns raised by Guerra et al. [13].

4.3. Preprocessing Ablation Analysis

Table 3 presents the cumulative impact of each preprocessing stage on SDLE accuracy, with Figure 3 providing a visual representation. The total preprocessing contribution of 7.43 percentage points (91.24% → 98.67%) exceeds the contribution of model architecture choices, underscoring that data preparation is at least as consequential as model design in practical IDS development.

Table 3. *Ablation Analysis: Cumulative Accuracy Impact of Preprocessing Stages.*

Preprocessing Stage	Accuracy (%)	Incremental Gain (%)
Raw Data (no preprocessing)	91.24	—
+ Label Encoding	93.47	+2.23
+ One-Hot Encoding	94.12	+0.65

Preprocessing Stage	Accuracy (%)	Incremental Gain (%)
+ Feature Selection (156 → 45 features)	96.41	+2.29
+ SMOTE Class Balancing	97.89	+1.48
+ Standard Scaling (Full Pipeline)	98.67	+0.78

Feature selection (+2.29%) and Synthetic Minority Oversampling Technique (SMOTE) class balancing (+1.48%) are the two largest individual contributors to the total gain of +7.43 percentage points. Feature selection is the single most impactful preprocessing step (+2.29%), reducing the feature space from 156 to 45 dimensions and eliminating noisy variables that cause overfitting. SMOTE balancing contributes a further 1.48 percentage points, predominantly by improving recall on minority attack classes, a critical operational requirement for security-critical IDS deployments where missed attacks carry higher costs than false positives. Standard scaling, whilst contributing the smallest individual increment (+0.78%), proved essential for training stability: normalization reduced average convergence from 95 to 72 epochs and decreased gradient truncation incidents from 12% to 2%.

4.4. Comparison with Some Related Works

Table 4 benchmarks the SDLE against recent IDS studies from the literature review. The SDLE achieves the highest accuracy and F1-score among studies evaluated on contemporary IoT datasets. The comparison with Alotaibi and Ilyas [5], also evaluated on ToN_IoT, demonstrates a direct 0.04-percentage-point accuracy improvement whilst additionally addressing multi-class per-attack evaluation that the prior study did not report.

Table 4. *Comparative Performance Against State-of-the-Art IDS Studies.*

Study	Architecture	Dataset	Accuracy (%)	F1 (%)
[5]	RF+KNN+LR+DT Stack	ToN_IoT	98.63	98.61
[9]	LSTM+CNN+RNN+GRU+DNN	CIC/SDN	~99.80	—
[18]	DNN+LSTM+GRU	CICIDS2017/CIDDS	98.09	98.09
[3]	KNN+SVM+RF → XGBoost	NSL-KDD	—	98.24
[12]	Seq. DNN + ETC	UNSW-NB15	97.93	97.00
[23]	KNN+NB+DT → Meta	NSL-KDD	99.01	—
SDLE (Proposed)	RNN+GRU+AE → LSTM	ToN_IoT	98.67	98.68

The balanced precision of 98.91% and recall of 98.45% give a false positive rate of 1.28% and a false negative rate of 1.55% on the 461,043-sample test set, that is 3,825 false positives and 2,611 false negatives. This balance matters in production. High precision lowers alert fatigue for analysts, a problem that Olasehinde [23] and Wazirali [30] tie to operational cost, while high recall keeps attacks from being missed. The inference throughput of 37,178 samples per second, equivalent to 0.027 ms of model computation per sample and 2.3 ms end-to-end once preprocessing is included, supports real-time, high-throughput IoT monitoring and is competitive with the efficiency goals set out in [4, 30]. Section 4.6 examines this cost in detail.

4.5. Statistical Validation of Performance Differences

The performance gaps reported in Table 1 are small in absolute terms, so their statistical reliability was assessed rather than assumed. Five-fold stratified cross-validation supplied five accuracy estimates per model, from which a 95% confidence interval was derived for each model and Welch's t-test was applied to every SDLE-versus-baseline pair, following the guidance of Demšar [10] on the comparison of classifiers. Table 5 reports the outcome, with Cohen's d included as a measure of effect size.

Table 5. Statistical Comparison of SDLE Against the Baseline Models (five-fold cross-validation, n = 5).

Comparison	Mean difference (%)	t	p-value	Cohen's d
SDLE vs RNN	+1.21	10.43	< 0.001	6.60
SDLE vs GRU	+0.78	8.06	< 0.001	5.10
SDLE vs Autoencoder	+1.94	13.05	< 0.001	8.25
SDLE vs Voting Ensemble	+0.99	9.15	< 0.001	5.79

Every comparison is significant at $p < 0.001$, and the effect sizes are large by any conventional threshold. The 95% confidence interval for SDLE accuracy is [98.52, 98.82], which does not overlap the interval of any baseline: RNN [97.17, 97.75], GRU [97.67, 98.11], Autoencoder [96.35, 97.11], and the voting ensemble [97.42, 97.94]. The SDLE also records the narrowest interval of the five models, ±0.15 percentage points against ±0.38 for the Autoencoder, which indicates that the ensemble is not merely more accurate but more stable across folds. The improvement is therefore attributable to the architecture rather than to a favourable partition of the data.

4.6. Computational Complexity and Deployment Feasibility

Detection quality alone does not establish that a model can be deployed on IoT infrastructure, so the cost of the ensemble was measured directly. Table 6 reports the parameter count, training cost, convergence behaviour, and memory footprint of each component.

Table 6. Computational Cost of the SDLE Components.

Component	Trainable parameters	Training time (h)	Epochs to converge	Runtime memory (MB)
RNN base learner	16,897	3.2	72	124
GRU base learner	12,289	2.6	68	98
Autoencoder base learner	2,885	1.8	65	45
LSTM meta-learner	9,409	0.9	58	38
SDLE (complete ensemble)	41,480	8.5	—	305

The complete ensemble holds 41,480 trainable parameters. Its asymptotic cost is dominated by the recurrent base learners, each of which requires $O(T(dh + h^2))$ multiply-accumulate operations per sequence for input dimension d, hidden width h, and T time steps. With $d = 45$, $h = 64$, and $T = 1$, the per-sample cost remains in the low tens of thousands of operations. The Autoencoder adds $O(dh)$, and the meta-learner operates on a three-element vector, so its cost is negligible. Inference over the full 461,043-sample test set completed in 12.4 seconds, giving a throughput of 37,178 samples per second, equivalent to 0.027 ms of model computation per sample and 2.3 ms end to end once preprocessing is included. Training the complete ensemble took 8.5 hours on the RTX 3080.

The memory figures in Table 6 record the runtime footprint of each loaded model inside the TensorFlow session rather than the size of the weights alone, which account for well under 1 MB; the complete system, including the Python runtime and the TensorFlow libraries, requires approximately 1.1 GB. For deployment, the operative figure is the 305 MB inference footprint, which sits within the memory available on commodity IoT gateway hardware such as a 4 GB Raspberry Pi 4 or an NVIDIA Jetson Nano, so the ensemble does not require a centralised server.

Two qualifications apply. The reported throughput and latency were obtained on a desktop GPU, and on a CPU-only gateway the same computation will be markedly slower. Although the parameter count is small enough that real-time operation remains plausible at typical IoT traffic volumes, this has not been measured here and is stated as an untested expectation rather than a result. The 8.5-hour training cost also means that retraining is a periodic offline operation rather than a continuous one. Post-training INT8 quantisation and structured pruning are the obvious routes to reducing the footprint

further and are left to future work. Compared with transformer-based intrusion detection such as RTIDS [32], whose self-attention layers scale quadratically in sequence length and which carries a considerably larger parameter budget than the recurrent components used here, the SDLE trades a degree of representational capacity for a footprint compatible with the constrained hardware it is intended to protect.

4.7. Generalisation, Explainability, and Deployment Considerations

Three constraints bound the claims made in this paper and are stated explicitly. The first, and the most consequential, is that the evaluation rests on a single dataset. ToN_IoT is the most contemporary IoT benchmark available and was selected for that reason, but a model tuned and tested on one corpus provides no evidence about its behaviour on traffic drawn from a different distribution, and accuracy reported on a single benchmark is a weak predictor of field performance [13]. The figure of 98.67% should therefore be read as performance on ToN_IoT and not as a general capability claim. Cross-dataset validation on CICIDS2017 and UNSW-NB15 is the priority extension, and it is not simply a matter of rerunning the existing code: those corpora expose different feature schemas, so the pipeline first requires a common feature representation before the base learners can be transferred, and both a zero-shot transfer test and a fine-tuned test would be needed to separate architectural generality from dataset-specific fitting.

The second constraint is explainability. The SDLE stacks four neural components, and an analyst who receives an alert cannot presently ask why it fired. This limits operational trust and creates difficulty in regulated environments where an adverse decision must be justified. SHAP [19] applied to the meta-feature vector would attribute each decision to the contributing base learners at negligible cost, since the meta-learner accepts only three inputs, while attention over the input features would localise the traffic characteristics responsible for a detection. Both are compatible with the present architecture, and neither requires the base learners to be retrained.

The third concerns deployment. The model is static once trained, so its accuracy will decay as attack behaviour drifts away from the training distribution. No adversarial evaluation was performed, which means that robustness against inputs crafted specifically to evade the detector is unknown. Periodic retraining, online or continual learning, and adversarial training with FGSM or PGD examples are the indicated mitigations, and each is left to future work.

5. Conclusion

The rapid expansion of IoT and IIoT infrastructure across critical sectors has outpaced the protective capacity of legacy signature-based intrusion detection, which cannot identify attacks it has not previously catalogued. This study was motivated by three persistent weaknesses in the existing literature: continued reliance on outdated benchmarks that no longer reflect contemporary IoT traffic, the evaluation of individual deep learning models in isolation despite their complementary failure modes, and the limited investigation of learned meta-combination strategies for fusing architecturally diverse base learners. The objective was to design, implement, and evaluate a stacked deep learning ensemble that fuses supervised recurrent models with unsupervised reconstruction-based detection under a single learned meta-learner, and to quantify its behaviour on a modern IoT benchmark. To meet this objective, the SDLE framework combined three base learners with complementary inductive biases: a Recurrent Neural Network and a Gated Recurrent Unit for temporal sequence modelling, and an Autoencoder for unsupervised anomaly scoring, with a two-layer LSTM acting as the meta-learner over their concatenated outputs. The framework was trained and tested on a 461,043-record subset of the ToN_IoT dataset spanning nine contemporary attack categories [6], following a five-stage preprocessing pipeline of label encoding, feature selection, one-hot encoding, SMOTE class balancing, and standard scaling. Performance was established through five-fold stratified cross-validation across accuracy, precision, recall, F1-score, and AUC-ROC, supplemented by per-attack-type, confusion-matrix, threshold-sensitivity, and computational analyses.

The SDLE achieved 98.67% accuracy, 98.91% precision, 98.45% recall, 98.68% F1-score, and an AUC-ROC of 0.9954 on the held-out test set, surpassing the strongest individual base learner (GRU) by 0.78 percentage points and the simple voting ensemble by 0.99 points. Detection remained consistent across the full attack taxonomy, with F1-scores ranging from 97.51% for MITM, the most data-scarce category, to 99.14% for DDoS. The preprocessing ablation showed that data preparation alone contributed 7.43 percentage points of accuracy, with feature selection (+2.29) and SMOTE balancing (+1.48) the two most consequential steps. These results carry two practical implications. First, the false positive rate of 1.28% and false negative rate of 1.55%, at an inference throughput of 37,178 samples per second and roughly 2.3 ms end-to-end per-sample latency, describe a detector accurate enough to limit analyst alert fatigue while fast enough for inline monitoring of operational IoT networks. Second, the finding that preprocessing rivals architecture in its effect on accuracy indicates that practitioners should invest in data preparation as deliberately as in model selection. The study makes four contributions to knowledge. It provides empirical validation of an LSTM as a meta-learner for stacked deep learning ensembles in intrusion detection; it supplies a quantified, stage-by-stage preprocessing ablation for a deep learning IDS on a modern IoT dataset; it reports per-attack-type performance across all nine ToN_IoT categories rather than a single binary figure; and it couples detection accuracy with a deployment-oriented computational analysis. Benchmarked against the most directly comparable prior study on the same dataset [5], the SDLE improved accuracy by 0.04 points while additionally resolving the per-attack evaluation that the earlier

work omitted.

In practical terms, the framework is suited to real-time monitoring of IoT and IIoT gateways, to edge deployment given its 305 MB memory footprint, and to security-operations settings where its adjustable decision threshold allows a single model to serve both general monitoring and high-assurance critical-infrastructure contexts. The work has limitations. Evaluation was confined to ToN_IoT, so generalisation to other traffic distributions remains to be demonstrated; the MITM class contained only 1,043 instances, which widened the variance of its per-class metrics; the ensemble remains a black-box detector without integrated explanation; and the static training regime does not yet accommodate concept drift. Future work will therefore extend evaluation to CICIDS2017 and UNSW-NB15, reformulate the task for multi-class attack identification, integrate SHAP or attention-based explanation to support analyst trust and regulatory accountability, and investigate online and federated learning to sustain performance as attack patterns evolve and to preserve privacy across distributed IoT deployments.

Abbreviations

AE	Autoencoder
ANN	Artificial Neural Network
AUC-ROC	Area Under the Receiver Operating Characteristic Curve
CNN	Convolutional Neural Network
DDoS	Distributed Denial of Service
DL	Deep Learning
DNN	Deep Neural Network
DoS	Denial of Service
FN	False Negative
FP	False Positive
FPR	False Positive Rate
GRU	Gated Recurrent Unit
IDS	Intrusion Detection System
IIoT	Industrial Internet of Things
IoT	Internet of Things
KNN	K-Nearest Neighbour
LSTM	Long Short-Term Memory
MITM	Man-in-the-Middle
ML	Machine Learning
MSE	Mean Squared Error
NIDS	Network Intrusion Detection System
RF	Random Forest
RNN	Recurrent Neural Network
SDLE	Stacked Deep Learning Ensemble
SHAP	SHapley Additive exPlanations
SMOTE	Synthetic Minority Over-sampling Technique
SVM	Support Vector Machine
TN	True Negative
TP	True Positive
XSS	Cross-Site Scripting

Author Contributions

Danjuma Israel Haruna: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Software, Validation, Writing – original draft

Boniface Kayode Alese: Conceptualization, Supervision, Writing – review & editing

Gabriel Babatunde Iwasokun: Methodology, Supervision, Writing – review & editing

David Bamidele Adewole: Investigation, Visualization

Ibraheem Temitope Jimoh: Formal analysis, Validation

Victoria Ibiyemi Omoniyi: Investigation, Visualization

Mojisola Olajumoke Ogunseye: Data curation, Resources

Conflicts of Interest

The authors declare no conflicts of interest.

References

[1] Abdulkareem, A., Adeyemo, V. E. and Balogun, A. O. "Experimental analysis of network intrusion detection using ensemble machine learning and artificial neural networks," Applied Intelligence, 2024.

[2] Akintoye, S. B., Ogunleye, B., Adeyemo, V. E., Adekunle, A. L. and Ogundokun, R. O. "Network intrusion detection and classification system employing supervised machine learning approaches," Journal of Computer Science and Technology Studies, vol. 6, no. 1, pp. 45–62, 2024.

[3] Ali, H., Saqib, M. and Ahmad, T. "Stacking-based ensemble for network intrusion detection," Future Generation Computer Systems, vol. 140, pp. 12–24, 2023.

[4] Almiani, M., AbuGhazleh, A., Al-Rahayfeh, A., Atiewi, S. and Razaque, A. "Deep recurrent neural network for IoT intrusion detection system," Simulation Modelling Practice and Theory, vol. 101, 2020.

[5] Alotaibi, B. and Ilyas, M. "Ensemble machine learning based identification of intrusions in IoT devices," IEEE Access, vol. 11, pp. 71555–71567, 2023.

[6] Alsaedi, A., Moustafa, N., Tari, Z., Mahmood, A. and Anwar, A. "TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems," IEEE Access, vol. 8, pp. 165130–165150, 2020.

[7] Alsoufi, M. A., Siraj, M. M. and Ghaleb, F. A. "Anomaly-based intrusion detection model using deep learning for IoT networks," Computer Modeling in Engineering & Sciences, vol. 141, no. 1, pp. 823–845, 2024.

[8] Bingu, R. and Jothilakshmi, S. "Ensemble-based deep learning model for intrusion detection in cloud and software-defined network environments," Computers & Electrical Engineering, vol. 108, 2023.

[9] Chawla, N. V., Bowyer, K. W., Hall, L. O. and Kegelmeyer, W. P. "SMOTE: Synthetic minority over-sampling technique," Journal of Artificial Intelligence Research, vol. 16, pp. 321–357, 2002.

[10] Demšar, J. "Statistical comparisons of classifiers over multiple data sets," Journal of Machine Learning Research, vol. 7, pp. 1–30, 2006.

[11] Dietterich, T. G. "Machine-learning research: Four current directions," AI Magazine, vol. 18, no. 4, pp. 97–136, 1997.

[12] Farhan, A., Ahmed, S. and Rahman, M. "Deep learning with feature selection for intrusion detection on UNSW-NB15," Neural Computing and Applications, vol. 37, no. 1, pp. 45–59, 2025.

[13] Guerra, J. L., Catania, C. and Veas, E. "Datasets are not enough: Challenges and directions in extracting intrusion detection benchmarks," arXiv preprint arXiv: 2108.08691, 2021.

[14] Gyimah, K., Opoku, D. and Frimpong, E. "Ensemble learning for intrusion detection systems: A survey," IEEE Access, vol. 12, pp. 11234–11251, 2024.

[15] Halbouni, A., Gunawan, T. S., Habaebi, M. H., Halbouni, M., Kartiwi, M. and Ahmad, R. "CNN-LSTM: Hybrid deep neural network for network intrusion detection system," IEEE Access, vol. 10, pp. 99837–99849, 2022.

[16] "Hybrid LSTM-Autoencoder and Random Forest cybersecurity framework with blockchain and Kafka integration," International Journal of Information Technology, Springer, 2025.

[17] Isife, K. I., Udanor, C. N. and Inyiama, H. C. "Deep learning for network intrusion detection using CICIDS2017 and CIDDS," Applied Intelligence, vol. 53, no. 7, pp. 8203–8218, 2023.

[18] Jabbar, M. A., Aluvalu, R. and Reddi, S. S. "RFAODE: A novel ensemble intrusion detection system," Procedia Computer Science, vol. 115, pp. 226–234, 2017.

[19] Lundberg, S. M. and Lee, S.-I. "A unified approach to interpreting model predictions," in Advances in Neural Information Processing Systems 30, pp. 4765–4774, 2017.

[20] Mienye, I. D. and Jere, N. "Deep learning for network intrusion detection: A review," Electronics, vol. 13, no. 7, p. 1263, 2024.

[21] Mienye, I. D. and Sun, Y. "A survey of ensemble learning: Concepts, algorithms, applications and prospects," IEEE Access, vol. 11, pp. 10048–10069, 2023.

[22] Moustafa, N. and Slay, J. "UNSW-NB15: A comprehensive data set for network intrusion detection systems," in Proc. Military Communications and Information Systems Conference (MilCIS), IEEE, 2015.

[23] Olasehinde, O. "Stacked ensemble intrusion detection approach for the protection of information systems," International Journal of Computer Applications, vol. 176, no. 38, pp. 9–17, 2020.

[24] Ravi, V., Chaganti, R. and Alazab, M. "Recurrent deep learning-based feature fusion ensemble meta-classifier approach for intelligent network intrusion detection system," Computers and Electrical Engineering, vol. 102, 2022.

[25] Sharif, A. and Ahmed, M. "Cyber intrusion detection by ensemble classifier using feature subset selection," in Proc. IEEE International Conference on Electro Information Technology (eIT), 2022.

[26] Tavallaee, M., Bagheri, E., Lu, W. and Ghorbani, A. A. "A detailed analysis of the KDD Cup 99 data set," in Proc. IEEE Symposium on Computational Intelligence for Security and Defense Applications, 2009.

[27] Thockchom, N., Singh, M. M. and Nandi, U. "A novel ensemble learning-based model for network intrusion detection," Complex & Intelligent Systems, vol. 9, no. 5, pp. 5665–5690, 2023.

[28] Ullah, I. and Mahmoud, Q. H. "Design and development of RNN anomaly detection model for IoT networks," IEEE Access, vol. 10, pp. 62722–62750, 2022.

[29] Waad, E. and Imad, B. "Deep learning model for intrusion detection system using CNN and Naive Bayes," International Journal of Computing and Digital Systems, vol. 13, no. 1, pp. 765–774, 2023.

[30] Wazirali, R. "An improved intrusion detection system based on KNN hyperparameter tuning and cross-validation," Arabian Journal for Science and Engineering, vol. 45, no. 12, pp. 10859–10873, 2020.

[31] Wolpert, D. H. "Stacked generalization," Neural Networks, vol. 5, no. 2, pp. 241–259, 1992.

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