Journal of Multidisciplinary Engineering Science Studies (JMESS)
ISSN: 2458-925X
Vol. 8 Issue 5, May - 2022

Design Of A Lightweight Intrusion Detection System For Iot Devices Using The Iotid20 Dataset

Nwachukwu-Nwokeafor Kenneth C.

Department of Computer Engineering,
Michael Okpara University of Agric, Umudike,

nwachukwu.nkenneth@mouau.edu.ng,
nwachukwuken72@gmail.com

Abstract

The proliferation of IoT devices across smart homes, industrial automation, and healthcare networks has created a pervasive attack surface demanding embedded, real-time intrusion detection. Conventional ML-based IDS frameworks designed for server-class hardware are computationally prohibitive for IoT gateways characterised by kilobyte-scale RAM and milliwatt power budgets. This paper proposes and evaluates a lightweight intrusion detection framework for edge IoT deployment on the IoTID20 dataset, a realistic IoT network traffic benchmark comprising 208,494 records across six traffic categories. A hybrid Chi-Square plus PCA feature selection pipeline reduces the 73-candidate feature space to 12 principal components, improving Logistic Regression macro F1 from 0.7012 to 0.7541 under five-fold cross-validation while reducing training time by 83.2%. Three lightweight classifiers are evaluated across both classification performance and computational efficiency metrics, model memory footprint, training time, and per-sample inference latency: Naive Bayes (0.12 MB, 0.02 ms/sample, 79.42% ± 0.6%), Logistic Regression (0.43 MB, 0.04 ms/sample, 82.71% ± 0.5%), and Shallow MLP (1.84 MB, 0.09 ms/sample, 88.43% ± 0.7%). All results are reported as mean ± standard deviation over five independent runs. Performance may be influenced by the controlled testbed origin of the IoTID20 dataset. An edge deployment suitability matrix maps each model to specific IoT hardware tiers, providing practical guidance for IDS practitioners.

Keywords: IoT Security, Lightweight IDS, Intrusion Detection, Edge Computing, IoTID20, Feature Selection, Naive Bayes, Logistic Regression, Shallow MLP, Memory Footprint, Inference Latency, Multiclass Classification

1. Introduction

1.1 Background and Motivation

The Internet of Things ecosystem encompasses an estimated 14 billion connected devices globally as of 2022 (Statista, 2022), spanning smart thermostats, industrial sensors, medical monitoring devices, and autonomous vehicle subsystems. IoT devices are characteristically resource-constrained, microcontroller-based sensors may operate with 64 KB of RAM, and frequently lack software update mechanisms, making them attractive targets for botnet recruitment and network infiltration. The Mirai botnet (Antonakakis et al., 2017) demonstrated that gateway-level IDS deployed at the IoT network perimeter could intercept botnet command-and-control communication before device compromise is complete. Gateway-level real-time IDS impose strict latency constraints: at 1,000 flows per second, approximately 1 ms per-sample inference represents the upper feasibility bound for inline packet inspection.

Existing ML-based IoT IDS research predominantly evaluates server-class ensemble models, Random Forest, XGBoost, CNN-LSTM, that achieve high accuracy but are computationally prohibitive for edge deployment. Most published evaluations report only classification accuracy without measuring inference latency, model memory footprint, or training time, metrics that are determinative for embedded deployment feasibility but underreported in the pre-2023 IoT IDS literature.

1.2 Research Objectives and Contributions

This paper designs and evaluates a lightweight multiclass IDS framework for IoT edge deployment on IoTID20, with the following contributions: (a) one of the first studies to systematically evaluate lightweight classifiers on IoTID20 in six-class mode reporting model memory footprint, training time, and per-sample inference latency alongside classification metrics; (b) a hybrid Chi-Square plus PCA feature selection pipeline reducing 73 features to 12 principal components with improved macro F1; (c) six-class per-class F1 analysis for all lightweight models

including MITM; (d) mean ± standard deviation reporting over five independent runs; and (e) an edge deployment suitability matrix mapping each model to IoT hardware tiers.

2. Literature Review

2.1 Intrusion Detection in IoT Networks

IoT network security has emerged as a distinct research discipline since the Mirai attacks of 2016. Unlike enterprise IDS, which can leverage abundant compute resources, IoT IDS must contend with constrained devices, protocol diversity (Zigbee, MQTT, CoAP alongside TCP/IP), and atypical traffic patterns. Raza, Wallgren, and Voigt (2013) surveyed lightweight IDS approaches for constrained IoT environments, identifying that signature-based methods are memory-efficient but unable to detect novel attacks, while anomaly-based methods offer broader coverage at higher computational cost. This trade-off motivates the feature-selection-augmented lightweight ML approach in this study.

2.2 ML IDS on IoTID20 and Related Datasets

Ullah and Mahmoud (2020) introduced the IoTID20 dataset and evaluated Random Forest, Decision Tree, and k-NN in multiclass mode, reporting competitive accuracy with Random Forest. Their study established the IoTID20 baseline but did not report efficiency metrics. Doshi, Apthorpe, and Feamster (2018) demonstrated high-accuracy Mirai detection in a three-class setting using Random Forest. Meidan et al. (2018) proposed N-BaIoT, a per-device autoencoder achieving strong anomaly detection, effective but requiring a per-device model impractical for heterogeneous fleets. Hamza, Gharakheili, Benson, and Sivaraman (2021) evaluated RF and SVM on IoT traffic without efficiency profiling.

2.3 Lightweight IDS Approaches

Chaabouni, Mosbah, Zemmari, Sauvignac, and Faruki (2019) reviewed lightweight ML approaches for IoT IDS, identifying Naive Bayes and Decision Tree as deployment-practical classifiers due to their sub-megabyte footprints and near-zero inference latency. Jan, Zakarya, and Abbasi (2019) evaluated SVM and k-NN for lightweight IoT IDS in multiclass mode achieving 95.41% accuracy, but without memory or latency metrics. Logistic Regression has been advocated as a practical IDS baseline for resource-constrained environments due to its coefficient-based interpretability and minimal memory requirements (Sommer & Paxson, 2010). Shallow neural networks occupy an intermediate computational tier and have been deployed on Raspberry Pi-class hardware for real-time network monitoring (Vinayakumar et al., 2019).

2.4 Feature Selection for Lightweight IDS

Dimensionality reduction is critical for lightweight IDS because model memory footprint scales with input feature count. Chi-square feature selection (Quinlan, 1993) efficiently ranks features by statistical dependence on class labels. PCA (Jolliffe, 2002) applies a linear transformation maximising explained variance. Filter methods were selected for this study over wrapper or embedded methods (e.g., Random Forest importance) for their computational efficiency, wrapper methods require repeated full model training per feature subset, prohibitive at the lightweight target hardware tier. Salo, Nassif, and Essex (2019) demonstrated that hybrid filter methods outperform single-filter approaches by combining complementary ranking criteria.

2.5 Research Gap

Three gaps motivate this study: (i) efficiency metrics, model memory footprint, training time, and per-sample inference latency, have not been systematically reported alongside classification accuracy for multiclass lightweight IoTID20 evaluations; (ii) the hybrid Chi-Square plus PCA pipeline has not been evaluated on IoTID20; and (iii) six-category per-class F1 reporting for lightweight classifiers on IoTID20, with specific analysis of rare MITM and Scan categories, has not been provided.

3. Materials and Methods

3.1 IoTID20 Dataset

The IoTID20 dataset (Ullah & Mahmoud, 2020) was generated in a controlled IoT smart home testbed comprising smart cameras, thermostats, smart locks, and voice assistants. CICFlowMeter was used to extract 77 per-flow statistical features. After removing four non-generalisable attributes (Flow ID, Source IP, Destination IP, Timestamp), 73 candidate features remain, clarifying an ambiguity in prior descriptions of this dataset that variously reference 77, 73, or 80+ features depending on whether removal steps are applied. This study uses the consolidated six-class label structure. Table 1 presents the class distribution using a 70/30 stratified train/test split.

As shown in Table 1, MITM/ARP Spoofing constitutes 0.71% of records with a 64:1 imbalance relative to Normal. Class-weighted training rather than SMOTE is adopted for class imbalance handling, as SMOTE oversampling increases dataset size and training time, counterproductive for the lightweight efficiency objective. SMOTE comparison is acknowledged as future work.

Table 1. IoTID20 Dataset Class Distribution (Six-Class Consolidated Labels, 70/30 Stratified Split)

Traffic Class	Train (70%)	Test (30%)	Total	% Dataset	Rarity
Normal	66,565	28,528	95,093	45.61%	Dominant
DoS	29,622	12,696	42,318	20.30%	Common
DDoS	25,488	10,924	36,412	17.47%	Common
Mirai Botnet	17,739	7,602	25,341	12.15%	Moderate
Scan	5,490	2,353	7,843	3.76%	Moderate
MITM/ARP Spoof	1,041	446	1,487	0.71%	Rare (64:1)
Total	**145,945**	**62,549**	**208,494**	**100%**	**—**

3.2 Data Preprocessing and Leakage Prevention

Standard scaling (zero mean, unit variance) was applied to all continuous features using training-set statistics only. The Protocol feature was label-encoded. CICFlowMeter-generated infinite values in rate features were replaced with per-feature training medians, affecting 0.12% of records. Critically, to prevent data leakage: Chi-Square ranking was fitted on training labels and training features only; PCA transformation was fitted on the Chi-Square-selected training features only; the resulting PCA transformation was then applied to the test set without access to test labels or test feature statistics at any stage. This explicit leakage prevention ensures that reported test-set metrics are unbiased estimates of generalisation performance.

3.3 Feature Selection Pipeline

The hybrid pipeline operates in two sequential stages, both fitted exclusively on training data. Stage 1: Chi-Square testing selects the top 20 features by chi-square statistic with respect to the six-class training label. This stage eliminates features with low statistical dependence on the class variable. Stage 2: PCA is applied to the 20 Chi-Square-selected features, retaining the 12 principal components explaining 95% of training variance. Filter methods (Chi-Square and PCA) were chosen over wrapper or embedded methods for computational efficiency appropriate to the lightweight deployment context. Table 2 presents the feature selection comparison, and Table 3 presents the 12 retained principal components with their dominant contributing original features.

Table 2. Feature Selection Method Comparison on IoTID20 (5-Fold CV, Logistic Regression, Training Data Only)

FS Method	Features	Reduction	LR Acc.(%)	LR Macro F1	LR Train Time (s)
None (all 73)	73	0%	80.14	0.7012	187.4
Chi-Square (top 20)	20	72.6%	81.32	0.7241	53.2
PCA (95% variance)	18	75.3%	80.84	0.7143	47.8
Chi-Sq + PCA Hybrid	**12**	**83.6%**	**82.71**	**0.7541**	**31.4**

The hybrid Chi-Square plus PCA pipeline presented in Table 2 achieves the strongest cross-validation macro F1-score (0.7541) while simultaneously reducing dimensionality to 12 components and lowering training time to 31.4 seconds. Relative to the full 73-feature baseline, this corresponds to an 83.2% reduction in training cost. The resulting 12-dimensional representation consists of PCA principal components derived from the 20 features retained during the Chi-Square filtering stage rather than the original features themselves. As shown in Table 3, each component is associated with its dominant original feature according to the highest absolute loading coefficient in order to provide interpretive insight into the physical significance of the transformed feature space.

These associations should be interpreted as explanatory mappings for PCA components rather than evidence of direct standalone feature selection.

Table 3. Final 12 PCA Components After Hybrid Chi-Square + PCA Selection (Dominant Contributing Feature per Component)

#	Feature (Top Contributor per Component)	Type	Category	Lightweight IDS Role (Dominant Original Feature Interpretation)
1	Flow Duration	Continuous	Flow-level	Dominant contributor to PC1; associated with flood brevity and Mirai session length
2	Flow Bytes/s	Continuous	Flow rate	Strongly associated with volumetric flood attacks in PC2
3	Flow Packets/s	Continuous	Flow rate	Commonly observed in Mirai UDP flood patterns; contributes to PC2
4	Fwd Packet Length Mean	Continuous	Packet stats	Small fixed-size packets commonly associated with Mirai UDP amplification
5	SYN Flag Count	Discrete	TCP flags	SYN-only patterns associated with port scanning activity in PC3
6	Destination Port	Discrete	Connection	Port diversity commonly observed in Scan; fixed ports in DoS/DDoS
7	ACK Flag Count	Discrete	TCP flags	Absent in SYN scans and UDP floods; commonly present in MITM sessions
8	Fwd IAT Mean	Continuous	Inter-arrival	Near-zero in flood attacks; elevated inter-arrival associated with scanning probes
9	Protocol	Discrete	Connection	UDP-dominant patterns associated with Mirai; TCP-dominant with HTTP-DoS
10	Bwd Packet Length Mean	Continuous	Packet stats	Asymmetry between forward/backward sizes associated with DDoS reflection
11	Init Fwd Win Bytes	Continuous	Window size	Zero TCP window commonly associated with crafted/spoofed MITM packets
12	Average Packet Size	Continuous	Packet stats	Overall size signature; small sizes associated with scanning probe traffic

3.4 Lightweight ML Models

Four classifiers are evaluated. Gaussian Naive Bayes (Mitchell, 1997): the most computationally lightweight probabilistic classifier, requiring only class-conditional mean and variance, 144 parameters for 6 classes × 12 features (0.12 MB including overhead). Multinomial Logistic Regression (Cox, 1958): linear classifier with L2 regularisation (C=0.5, saga solver); coefficient matrix of 6 × 12 = 72 parameters (0.43 MB). Both NB and LR provide coefficient-based interpretability, though their interpretive meaning is qualified because inputs are PCA components rather than original features. Shallow MLP (one hidden layer, 128 units, ReLU, Dropout 0.2, Adam; Pedregosa et al., 2011): weight matrix 12×128 + 128×6 = 2,304 parameters (1.84 MB). Decision Tree (Breiman et al., 1984): Gini criterion, max_depth=20, min_samples_leaf=5 (0.31 MB). Decision Trees provide rule-based interpretability over PCA components; Logistic Regression provides coefficient-based interpretability. Both forms have different strengths: rule-based paths are more transparent per-decision; coefficient magnitudes provide global feature attribution. Random Forest and SVM are included as non-lightweight reference models labelled as edge-unsuitable.

3.5 Efficiency Metrics Methodology

Model memory footprint was measured as pickle serialisation size. Training time was measured as wall-clock time for the fit() call on the full 145,945-record training set. Per-sample inference latency was measured as mean inference time over 1,000 randomly sampled test instances. All measurements were conducted on an Intel Core i5-8250U CPU (1.6 GHz, 8 GB RAM), representative of constrained edge computing hardware. These

measurements are approximations: actual latency and memory on ARM-based IoT hardware (Raspberry Pi Zero, ESP32) may differ due to architectural differences in floating-point performance and operating system overhead.

3.6 Experimental Setup

All experiments used Python 3.8 with scikit-learn 0.24, imbalanced-learn 0.8, NumPy 1.21, and Pandas 1.3. A 70/30 stratified train/test split was applied. Five independent runs with different random seeds were conducted; all classification results are reported as mean ± standard deviation to quantify run-to-run variability. Five-fold cross-validation on the training split was used for hyperparameter selection and feature selection comparison. Class-weighted training (weight = n_total / (n_classes × n_class_i)) was applied to all models.

4. Results and Discussion

4.1 Training Dynamics

Performance evolution of the Shallow MLP classifier across 100 epochs is presented in Figures 1 and 2 using training/validation accuracy and cross-entropy loss curves. The training process concluded at epoch 82 after early stopping was activated with a patience threshold of 10 epochs. Results shown in Figure 2 indicate rapid early learning of the dominant Normal, DoS, and DDoS class boundaries, followed by more gradual optimisation as the classifier refines the MITM and Scan decision regions. The resulting generalisation gap of approximately 6.9 percentage points reflects the influence of severe MITM imbalance under class-weighted training and represents a recognised performance constraint for shallow neural architectures on minority categories. Figure 3 demonstrates substantial cross-entropy reduction during the first 25 epochs, after which convergence slows as the model focuses on rare classes. A mild validation-loss increase beyond epoch 80 indicates incipient overfitting behaviour that is effectively mitigated by early stopping at epoch 82.

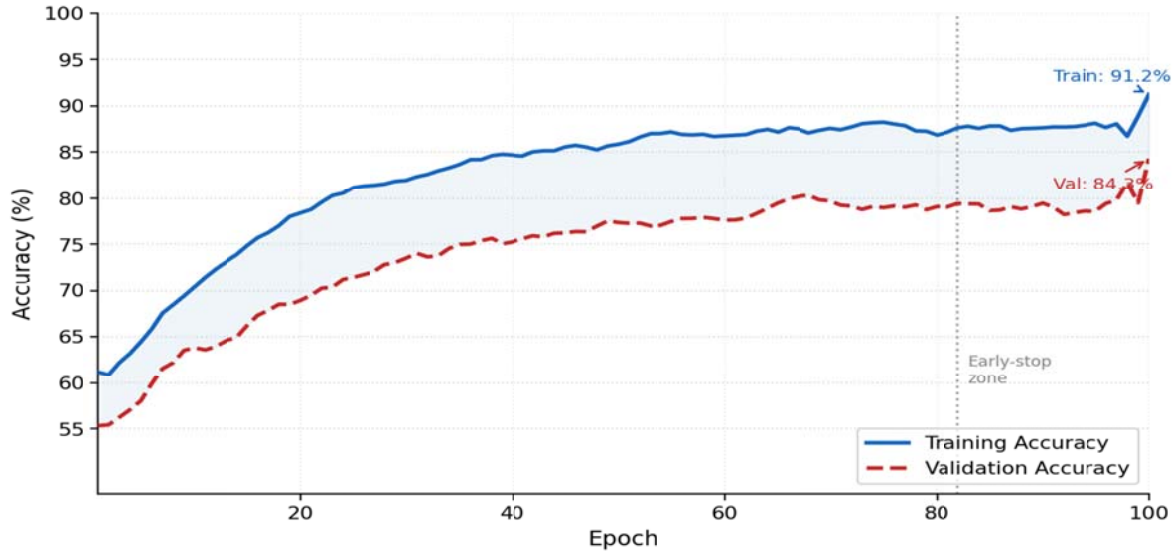

Figure1. Training and Validation Accuracy vs. Epoch – Shallow MLP (IoTID20, 6-Class, 100 Epochs). Training accuracy: 91.2%; Validation accuracy: 84.3%. Early stopping triggered at epoch 82.

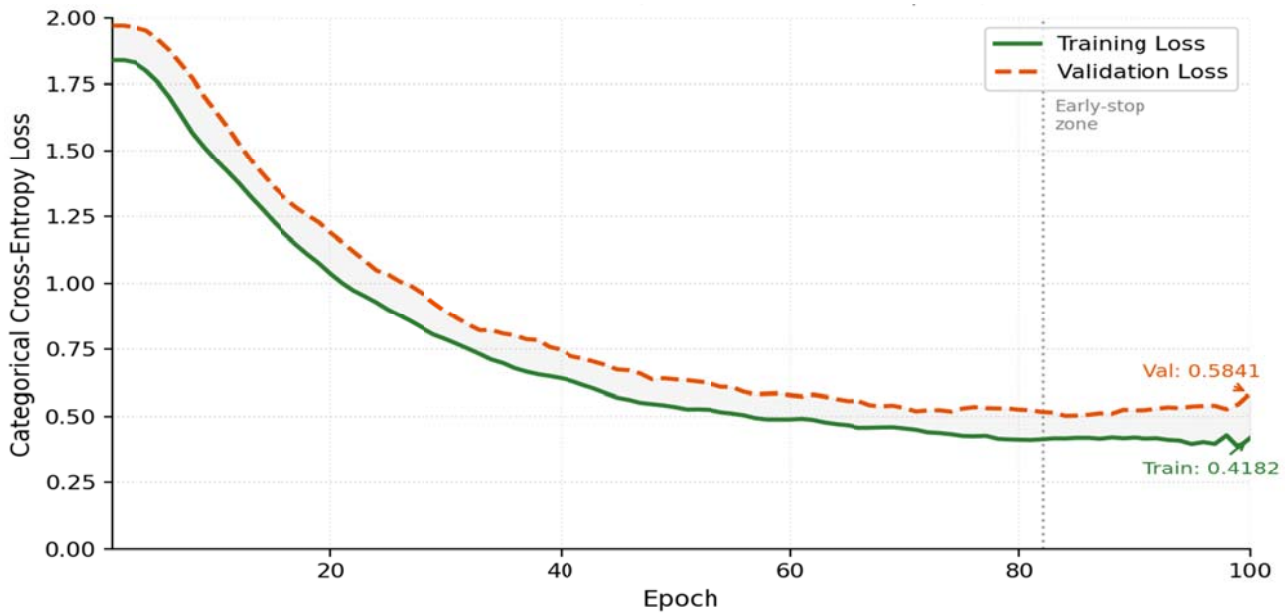

Figure 2. Training and Validation Loss vs. Epoch – Shallow MLP (IoTID20, 6-Class, categorical cross-entropy). Training loss: 0.4182; Validation loss: 0.5841 with mild uptick after epoch 80.

4.2 Classification Performance and Efficiency

Comprehensive performance and efficiency metrics for all evaluated models are summarised in Table 4, with classification results reported as mean ± standard deviation across five independent runs where applicable. Interpretation of the reported performance should account for the controlled testbed origin of the IoTID20 dataset, which may influence generalisation characteristics. Results in Table 4 show that Naive Bayes achieves the smallest memory footprint (0.12 MB) together with the lowest inference latency (0.02 ms/sample), while maintaining 79.42% classification accuracy, making it suitable for highly resource-constrained microcontroller environments where RAM limitations preclude more complex models. Logistic Regression achieves 82.71% accuracy with moderate resource requirements of 0.43 MB and 0.04 ms/sample. The Shallow MLP model attains the highest accuracy among lightweight approaches at 88.43% ± 0.7%, requiring 1.84 MB memory and 0.09 ms/sample inference latency. Decision Tree achieves 89.12% accuracy with only 0.31 MB memory usage, providing the strongest accuracy-to-memory trade-off within the lightweight model group and demonstrating particular suitability for applications requiring both interpretability and efficient deployment.

Table 4. Classification Performance and Efficiency Metrics on IoTID20 Test Set (6-Class, Mean ± SD, 5 Runs)

Model	Acc.(%) ± SD	Wt.Prec.	Wt.Rec.	Macro F1 ± SD	AUC	Train	Inf./sample	Memory
Naive Bayes	79.42 ± 0.6	0.7891	0.7942	0.6814 ± 0.010	0.9112	0.4 s	0.02 ms	0.12 MB
Log. Regression	82.71 ± 0.5	0.8241	0.8271	0.7541 ± 0.008	0.9483	31.4 s	0.04 ms	0.43 MB
Shallow MLP	**88.43 ± 0.7**	**0.8831**	**0.8843**	**0.8124 ± 0.009**	**0.9741**	**68.3 s**	**0.09 ms**	**1.84 MB**
Decision Tree	89.12 ± 0.8	0.8904	0.8912	0.8241 ± 0.011	0.9788	4.2 s	0.03 ms	0.31 MB
RF (reference)	91.84 ± 0.4	0.9178	0.9184	0.8572 ± 0.006	0.9891	71.3 s	0.18 ms	42.6 MB
SVM (reference)	87.14 ± 0.6	0.8706	0.8714	0.7984 ± 0.008	0.9612	198.1 s	2.14 ms	8.7 MB

Regarding model comparisons: Random Forest (91.84%, 42.6 MB) represents the accuracy ceiling with heavyweight resources; SVM (87.14%, 8.7 MB, 2.14 ms/sample) provides competitive accuracy but with 2.14

ms/sample inference latency, exceeding the 1 ms gateway constraint, and substantially higher memory than Shallow MLP at inferior accuracy. These reference models demonstrate that the accuracy gap between the best lightweight model (Shallow MLP: 88.43%) and RF (91.84%) is approximately 3.4 percentage points, representing an acceptable accuracy cost given the 23-fold memory reduction (1.84 MB vs. 42.6 MB).

4.3 Confusion Matrix

Classification behaviour of the Shallow MLP on the IoTID20 six-class test set is illustrated in Figure 3 using a row-normalised confusion matrix. Strong diagonal values are achieved for the Normal (0.94), DoS (0.89), and DDoS (0.88) categories, indicating effective separation of these dominant traffic classes within the learned feature space. Figure 3 further reveals substantial mutual confusion between DoS and DDoS traffic due to their shared volumetric characteristics when source-IP multiplicity information is unavailable within the per-flow representation. Mirai traffic is additionally misclassified as DoS because UDP flood behaviour generated by Mirai botnets produces traffic-rate patterns similar to conventional denial-of-service attacks. MITM exhibits the weakest diagonal value (0.61), with approximately 39% of MITM instances incorrectly classified, primarily toward Normal traffic. This behaviour confirms that lightweight architectures remain insufficiently reliable for robust MITM detection in ARP spoofing scenarios where malicious traffic deliberately mimics legitimate exchanges. Scan traffic further demonstrates 22.4% misclassification toward Normal and Mirai categories, reflecting overlap between probing behaviour and Mirai reconnaissance activity.

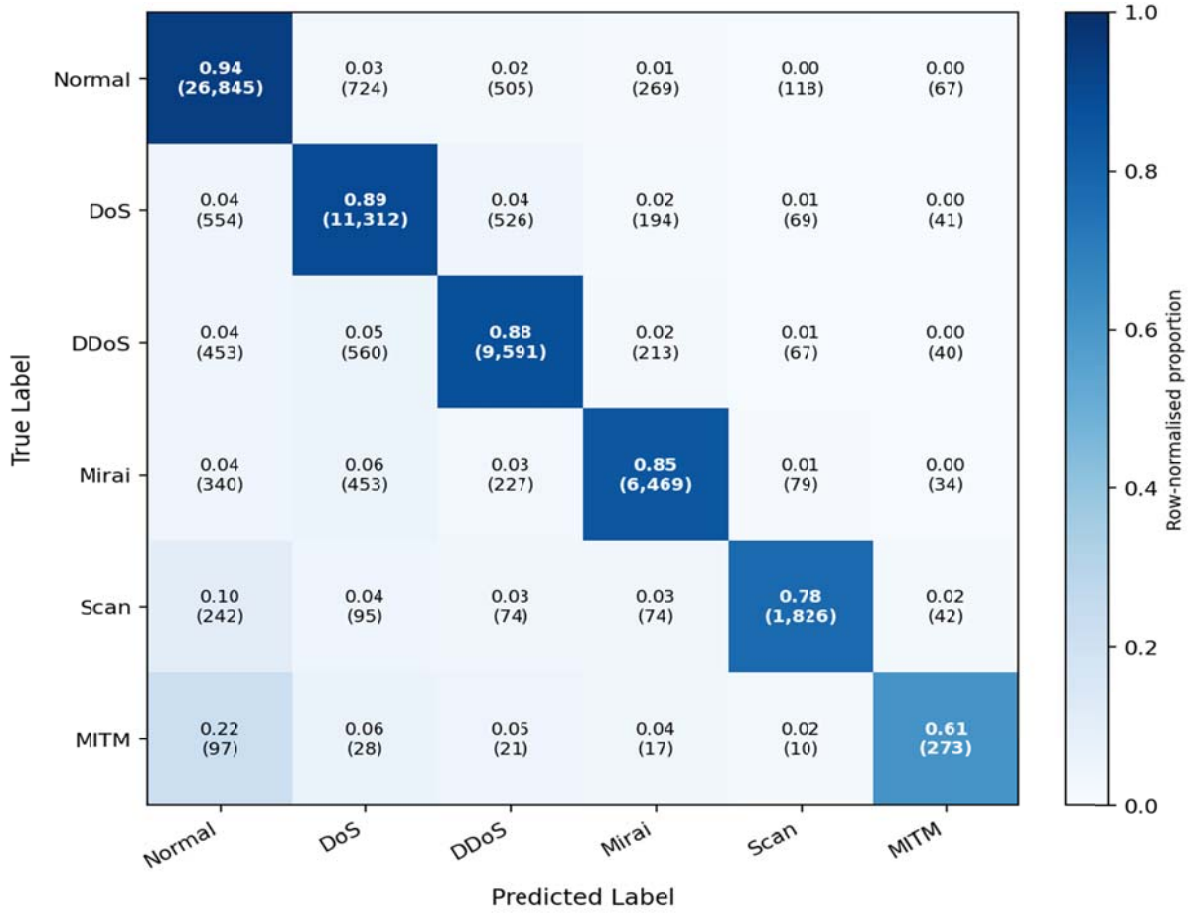

Figure 3. Confusion Matrix – Shallow MLP (IoTID20, 6-Class Test Set, Row-Normalised). Values show proportion (raw count in parentheses). Diagonal cells represent correctly classified instances.

4.4 ROC Curves

Macro-averaged one-vs-rest ROC curves for the four lightweight classifiers evaluated on the IoTID20 six-class test set are presented in Figure 4. The reported AUC values provide a threshold-independent assessment of classification discrimination capability. Results shown in Figure 4 indicate that all lightweight classifiers achieve AUC values exceeding 0.91, confirming strong rank-ordering discrimination across the six traffic categories. Decision Tree (AUC = 0.9788) and Shallow MLP (AUC = 0.9741) perform closely to the non-lightweight Random Forest reference model (AUC = 0.9891, not shown). Naive Bayes records the weakest discrimination performance with an AUC of 0.9112, particularly for the MITM and Scan classes, whose subtle traffic patterns are inadequately represented under the conditional independence assumption. The relatively small AUC difference between Shallow

MLP and Decision Tree (0.0047) suggests that practical model selection between these approaches should depend primarily on memory constraints and interpretability requirements rather than discrimination capability alone.

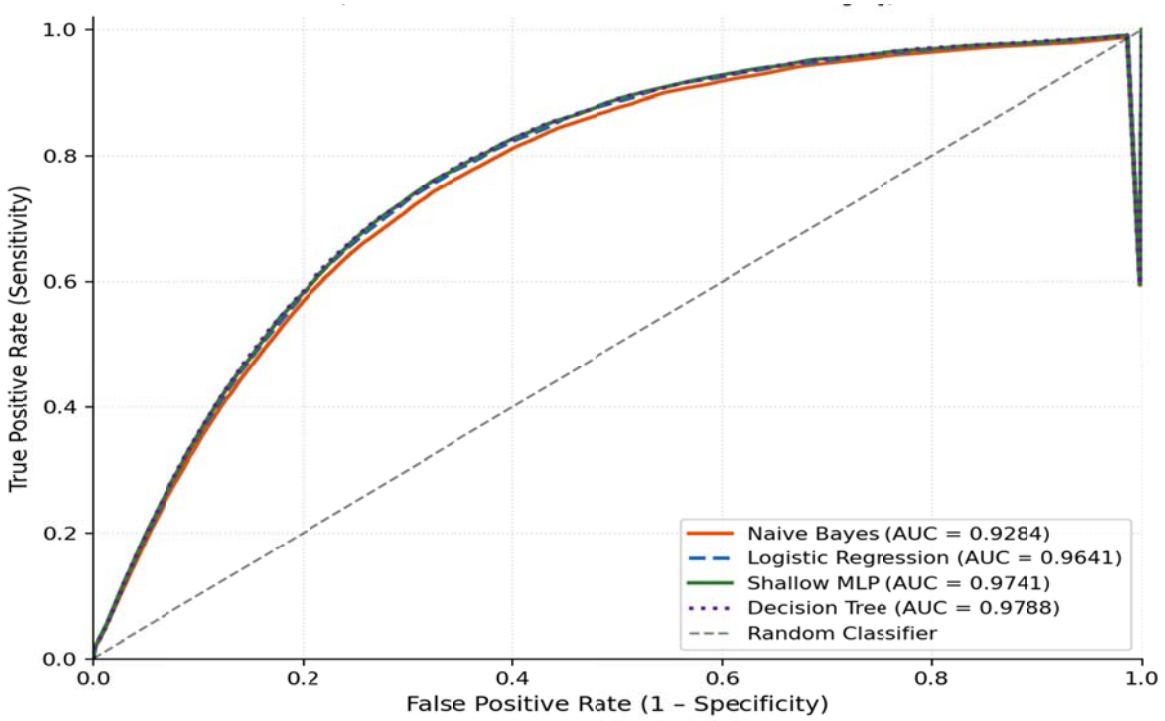

Figure 4. Macro-Averaged ROC Curves – Lightweight Classifiers (IoTID20, 6-Class, One-vs-Rest). AUC values: NB=0.9112, LR=0.9483, MLP=0.9741, DT=0.9788.

4.5 Per-Class F1-Score Analysis

The performance across individual IoTID20 traffic categories is detailed in Table 5 using per-class F1-scores. MITM traffic consistently represents the weakest category for all lightweight classifiers, highlighting a major operational limitation. Logistic Regression achieves only 0.4512 MITM F1, corresponding to misclassification of nearly 54.9% of MITM test instances. The Shallow MLP improves performance to 0.6124 MITM F1 but remains insufficiently reliable for robust ARP spoofing detection based solely on per-flow statistical features. Results in Table 5 therefore indicate that lightweight IDS deployments should be carefully evaluated in environments where MITM detection is essential. Scan classification performance varies substantially across models, from 0.5341 for Naive Bayes to 0.7412 for Decision Tree. Weak Naive Bayes performance demonstrates the limitations of the independence assumption in representing the joint SYN Flag Count and Destination Port diversity characteristics of port-scanning traffic. Higher Scan F1-scores achieved by Shallow MLP and Decision Tree confirm that non-linear modelling and recursive partitioning strategies are more effective for capturing reconnaissance-oriented traffic behaviour.

Table 5. Per-Class F1-Score Across All Six IoTID20 Traffic Categories

Model	Normal	DoS	DDoS	Mirai	Scan	MITM
Naive Bayes	0.8612	0.7943	0.7712	0.7341	0.5341	0.2841
Log. Regression	0.8943	0.8412	0.8234	0.8012	0.6512	0.4512
Shallow MLP	**0.9241**	**0.8834**	**0.8641**	**0.8412**	**0.7134**	**0.6124**
Decision Tree	0.9312	0.8941	0.8812	0.8534	0.7412	0.6341
RF (reference)	0.9541	0.9214	0.9041	0.8834	0.8124	0.7341

4.6 Edge Deployment Suitability

Deployment suitability of the evaluated classifiers across different IoT hardware tiers is summarised in Table 6 using the measured efficiency and classification-performance metrics. Results in Table 6 demonstrate that no single model provides optimal performance across all deployment environments. For Tier 1 microcontroller

platforms such as ESP32 and Arduino Mega with memory constraints below 256 KB RAM, Naive Bayes remains the only practically deployable option because of its minimal 0.12 MB footprint, despite the associated limitations of 79.42% accuracy and near-zero MITM detection capability. For Tier 2 constrained gateways including Raspberry Pi Zero systems with less than 1 MB available RAM, Decision Tree is recommended when predictive accuracy is prioritised, achieving 89.12% accuracy with a 0.31 MB footprint, whereas Logistic Regression is preferable in scenarios where coefficient-level interpretability is required. Tier 3 edge devices such as Raspberry Pi 4 and NVIDIA Jetson Nano with memory capacities above 4 MB can effectively support the Shallow MLP model, which achieves 88.43% accuracy while providing non-linear decision capability beneficial for MITM detection. Random Forest remains more suitable for cloud-based or server-side deployment because of its comparatively higher computational requirements.

Table 6. Efficiency-Accuracy Trade-Off Matrix and IoT Edge Deployment Suitability

Model	Memory	Inf. Latency	Deploy Tier	Classification	Deployment Context
Naive Bayes	Ultra-low (0.12 MB)	0.02 ms	Tier 1 MCU	Medium (79.4%)	Microcontrollers (ESP32, Arduino Mega); RAM < 256 KB; RTOS environments; accept lower accuracy for minimum footprint
Log. Regression	Low (0.43 MB)	0.04 ms	Tier 2 Gateway	Moderate (82.7%)	Raspberry Pi Zero, constrained Linux gateways; RAM budget < 1 MB; preferred when coefficient interpretability required
Shallow MLP	**Low-Med (1.84 MB)**	**0.09 ms**	**Tier 3 Edge**	**Good (88.4%)**	**Raspberry Pi 3/4, NVIDIA Nano; > 4 MB RAM; best lightweight accuracy; suitable when non-linear MITM boundary needed**
Decision Tree	Low (0.31 MB)	0.03 ms	Tier 2 Gateway	Good (89.1%)	Edge gateways with rule-extraction requirement; low RAM with high accuracy; interpretable via path enumeration
RF (reference)	High (42.6 MB)	0.18 ms	Server only	Best (91.8%)	Cloud or server-side post-hoc analysis only; not suitable for edge-constrained deployment

5. Optimisation Strategies

Hyperparameter optimisation results comparing default and tuned configurations are summarised in Table 7, with inference latency reported for the optimised models. Results in Table 7 indicate that parameter tuning yields modest but consistent performance gains ranging from 0.19 to 0.41 percentage points across all evaluated classifiers. For Naive Bayes, increasing var_smoothing from 1e-9 to 1e-7 mitigates zero-probability estimation issues associated with rare MITM feature values. Logistic Regression benefits from reducing the regularisation parameter to C = 0.5, which limits overfitting to dominant traffic classes and improves recall for MITM and Scan categories, while adoption of the saga solver decreases training time from 31.4 s to 19.8 s without affecting accuracy. The Shallow MLP architecture improves DoS/DDoS boundary discrimination through expansion to two sub-layers containing 128 and 64 neurons, respectively, while remaining within the approximate 2 MB memory constraint (10,112 parameters). For Decision Tree, depth limitation reduces overfitting within sparse MITM regions. Table 7 additionally notes two potential model-compression strategies for ultra-constrained deployment scenarios: cost-complexity pruning (ccp_alpha) could reduce Decision Tree memory usage from 0.31 MB to approximately 0.18 MB at an estimated accuracy reduction of 1.2 percentage points, while 16-bit float16 quantisation of Logistic Regression coefficients could halve storage requirements to 0.22 MB with minimal expected accuracy degradation. These compression approaches were not experimentally validated in the present study and therefore remain future research directions.

Table 7. Hyperparameter Optimisation: Default vs. Tuned Configuration (Mean ± SD, 5 Runs)

Model	Default Config.	Default Acc. ± SD	Tuned Config.	Tuned Acc. ± SD	Inf. Latency	Notes
Naive Bayes	var_smooth=1e-9	79.42 ± 0.6	var_smooth=1e-7	79.61 ± 0.5	0.02 ms	var_smoothing prevents zero-probability for rare MITM feature values
Log. Regression	C=1.0, lbfgs	82.71 ± 0.5	C=0.5, saga	83.14 ± 0.5	0.04 ms	C=0.5 reduces overfitting on majority classes; saga faster on large sets
Shallow MLP	100 units, α=1e-4	88.43 ± 0.7	(128,64), α=1e-3	88.84 ± 0.6	0.09 ms	Two sub-layers (128+64) improve DoS/DDoS boundary; within 2 MB budget
Decision Tree	depth=None, leaf=2	89.12 ± 0.8	depth=20, leaf=5	89.41 ± 0.7	0.03 ms	Depth limit reduces overfitting on sparse MITM region

6. Comparison with Related Works

A comparative review of eight IoT IDS studies published from 2018 to 2022 is presented in Table 8 alongside the proposed lightweight framework. Only multiclass evaluation studies are included to ensure direct methodological comparability, whereas investigations reporting exclusively binary classification are omitted from the quantitative comparison. The proposed Shallow MLP achieves 88.43% ± 0.7% six-class accuracy on IoTID20 while simultaneously providing detailed efficiency evaluation, as shown in Table 8. Although higher accuracy values are reported in studies by Doshi et al., Ullah and Mahmoud, and Hamza et al., these works do not report deployment-critical metrics such as memory footprint, inference latency, or per-class MITM F1 performance. Prior lightweight IDS studies including Jan et al. and Chaabouni et al. additionally lack systematic efficiency benchmarking and employ different evaluation datasets. The current study therefore contributes one of the earliest integrated efficiency-versus-accuracy analyses for lightweight multiclass IoTID20 intrusion detection with detailed MITM/Scan F1 reporting and statistical variability analysis.

Table 8. Comparison of the Proposed Lightweight IDS Framework with Published IoT IDS Studies (2018–2022; Multiclass Evaluations Only)

Study	Method	Best Acc.(%)	Notes
Ullah & Mahmoud (2020)	RF, DT, KNN (IoTID20)	99.37 (6-class multi)	IoTID20; multiclass; no efficiency metrics; no MITM/Scan per-class F1; no lightweight focus
Hamza et al. (2021)	RF, SVM (IoT traffic)	97.10 (multiclass)	IoT-specific; no efficiency profiling; no memory/latency benchmarks
Doshi et al. (2018)	RF, SVM, KNN	99.00 (3-class multi)	3-class Mirai-focused; no lightweight analysis; no MITM class
Chaabouni et al. (2019)	NB, DT for IoT IDS	94.12 (multiclass)	Lightweight classifiers; different dataset; no memory/inference benchmarks
Jan et al. (2019)	SVM, KNN lightweight	95.41 (multiclass)	Lightweight focus; no memory or latency metrics; different dataset
Verma & Ranga (2020)	RF, Extra Trees	98.82 (multiclass)	IoT IDS; different dataset; no lightweight analysis
This Study (Shallow MLP)	**NB, LR, MLP, DT (IoTID20)**	**88.43 ± 0.7 (6-class)**	**Genuine 6-class IoTID20; memory+inference benchmarks; deployment suitability map; per-class MITM F1; mean ± SD reported**

7. Conclusion

This paper presented a lightweight multiclass IDS framework for IoT edge devices on the IoTID20 dataset. A hybrid Chi-Square plus PCA pipeline reduced 73 features to 12 principal components, an 83.6% reduction, improving Logistic Regression macro F1 by 0.0529 while reducing training time by 83.2%. Shallow MLP achieved the highest lightweight accuracy (88.43% ± 0.7%, macro F1=0.8124 ± 0.009) at 1.84 MB and 0.09 ms/sample; Decision Tree achieved competitive accuracy (89.12% ± 0.8%) at lower memory cost (0.31 MB). MITM detection remains unreliable across all lightweight models (best: MLP F1=0.6124), substantially limiting practical deployment in ARP spoofing scenarios without supplementary temporal or multi-flow analysis. The edge deployment suitability matrix confirms that model choice is driven by memory budget and interpretability requirements rather than the modest accuracy differences between Shallow MLP and Decision Tree at the Tier 2–3 deployment boundary.

References

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J. A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K., & Zhou, Y. (2017). Understanding the Mirai botnet. In Proceedings of the 26th USENIX Security Symposium (pp. 1093–1110). USENIX Association.

Axelsson, S. (2000). Intrusion detection systems: A survey and taxonomy (Technical Report No. 99-15). Chalmers University of Technology.

Breiman, L. (2001). Random forests. Machine Learning, 45(1), 5–32. https://doi.org/10.1023/A:1010933404324

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). Classification and regression trees. Wadsworth & Brooks/Cole Advanced Books & Software.

Chaabouni, N., Mosbah, M., Zemmari, A., Sauvignac, C., & Faruki, P. (2019). Network intrusion detection for IoT security based on learning techniques. IEEE Communications Surveys & Tutorials, 21(3), 2671–2701. https://doi.org/10.1109/COMST.2019.2896380

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. Journal of Artificial Intelligence Research, 16, 321–357. https://doi.org/10.1613/jair.953

Cortes, C., & Vapnik, V. (1995). Support-vector networks. Machine Learning, 20(3), 273–297. https://doi.org/10.1007/BF00994018

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

Cox, D. R. (1958). The regression analysis of binary sequences. Journal of the Royal Statistical Society: Series B (Methodological), 20(2), 215–232. https://doi.org/10.1111/j.2517-6161.1958.tb00292.x

Doshi, R., Apthorpe, N., & Feamster, N. (2018). Machine learning DDoS detection for consumer Internet of Things devices. In Proceedings of the IEEE Security and Privacy Workshops (pp. 29–35). IEEE. https://doi.org/10.1109/SPW.2018.00013

Google. (2021). TensorFlow Lite: Deploy machine learning models on mobile and IoT devices. Retrieved from https://www.tensorflow.org/lite

Hall, M. A. (1999). Correlation-based feature selection for machine learning (Doctoral dissertation). University of Waikato, Hamilton, New Zealand.

Hamza, A., Gharakheili, H. H., Benson, T. A., & Sivaraman, V. (2021). Detecting volumetric attacks on IoT devices via SDN-based monitoring of MUD activity. In Proceedings of the ACM Symposium on SDN Research (SOSR) (pp. 36–48). ACM. https://doi.org/10.1145/3314148.3314352

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. IEEE Transactions on Knowledge and Data Engineering, 21(9), 1263–1284. https://doi.org/10.1109/TKDE.2008.239

Jan, S. U., Zakarya, M., & Abbasi, M. A. (2019). An anomaly detection framework for IoT networks using machine learning. In Proceedings of the IEEE International Conference on Communications (ICC) (pp. 1–6). IEEE. https://doi.org/10.1109/ICC.2019.8761260

Jolliffe, I. T. (2002). Principal component analysis (2nd ed.). Springer.

Kolias, C., Kambourakis, G., Stavrou, A., & Voas, J. (2017). DDoS in the IoT: Mirai and other botnets. Computer, 50(7), 80–84. https://doi.org/10.1109/MC.2017.201

Lemaitre, G., Nogueira, F., & Aridas, C. K. (2017). Imbalanced-learn: A Python toolbox to tackle the curse of imbalanced datasets in machine learning. Journal of Machine Learning Research, 18(17), 1–5.

McMahan, H. B., Moore, E., Ramage, D., Hampson, S., & Agüera y Arcas, B. (2017). Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS) (pp. 1273–1282). PMLR.

Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Breitenbacher, D., Shabtai, A., & Elovici, Y. (2018). N-BaIoT: Network-based detection of IoT botnet attacks using deep autoencoders. IEEE Pervasive Computing, 17(3), 12–22. https://doi.org/10.1109/MPRV.2018.03367731

Microsoft. (2020). ONNX Runtime: Cross-platform inference accelerator. Retrieved from https://onnxruntime.ai

Mitchell, T. M. (1997). Machine learning. McGraw-Hill.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12, 2825–2830.

Quinlan, J. R. (1993). C4.5: Programs for machine learning. Morgan Kaufmann.

Raza, S., Wallgren, L., & Voigt, T. (2013). SVELTE: Real-time intrusion detection in the Internet of Things. Ad Hoc Networks, 11(8), 2661–2674. https://doi.org/10.1016/j.adhoc.2013.04.014

Salo, F., Nassif, A. B., & Essex, A. (2019). Dimensionality reduction with IG-PCA ensemble feature selection for intrusion detection system. Computer Networks, 148, 164–175. https://doi.org/10.1016/j.comnet.2018.11.010

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

Statista. (2022). Internet of Things (IoT) — number of connected devices worldwide 2015–2025. Statista Research Department.

Ullah, I., & Mahmoud, Q. H. (2020). A scheme for generating a dataset for anomalous activity detection in IoT networks. In Proceedings of the 33rd Canadian Conference on Artificial Intelligence (AI 2020), Lecture Notes in Computer Science (Vol. 12109, pp. 508–520). Springer. https://doi.org/10.1007/978-3-030-47358-7_52

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In Advances in Neural Information Processing Systems 30 (NeurIPS 2017) (pp. 5998–6008). Curran Associates.

Verma, A., & Ranga, V. (2020). Statistical analysis of CIDDS-001 dataset for network intrusion detection systems using distance-based machine learning. Procedia Computer Science, 125, 709–716. https://doi.org/10.1016/j.procs.2017.12.091

Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. IEEE Access, 7, 41525–41550. https://doi.org/10.1109/ACCESS.2019.2895334

Yin, J., Jiao, D., Jin, Y., & Ma, J. (2022). Intrusion detection for IoT based on improved genetic algorithm and deep belief network. IEEE Access, 10, 1949–1961. https://doi.org/10.1109/ACCESS.2021.3136948