

ANOMALY DETECTION IN INTERNET OF THINGS NETWORKS USING EXPLAINABLE DEEP LEARNING TECHNIQUES

Ezeibeau Ozioma Stephanie, Mgbеafulike Ike J., Okeke Ogochukwu C.

Department of Computer Science, Chukwuemeka Odumegwu Ojukwu University, Uli, Anambra State

¹ezeibeaustephanie@gmail.com, ²ike.mgbеafulike@gmail.com, ³ogoookeke@yahoo.com

ABSTRACT

The proliferation of the Internet of Things (IoT) increases interconnectivity across homes, healthcare, manufacturing, transportation and other cyber-physical systems, while also expanding the attack surface of those networks. This paper reports an implemented and evaluated explainable deep-learning detector for anomalies in IoT network traffic. The study uses the CICIOT2023 corpus of the Canadian Institute for Cybersecurity. After cleaning, label encoding, z-score normalisation fitted on the training partition only, Extra Trees feature selection and training-set class balancing, a stacked Gated Recurrent Unit (GRU) classifier was trained on sliding windows of length 10. The trained network performs 34-class traffic attribution and is also mapped to a binary benign-versus-anomalous decision. SHapley Additive exPlanations (SHAP) were computed after training to obtain global feature rankings and local explanations of individual alerts. On the held-out test set the implemented GRU attained 97.56% accuracy, 97.41% weighted precision, 97.56% weighted recall, 97.42% weighted F1-score and a Cohen kappa of 0.9736. Family-level analysis shows that flooding classes are detected reliably, whereas reconnaissance, spoofing, web-based and brute-force traffic remain the hardest minority groups. The same train/test split was used to train Random Forest, deep neural network, convolutional and long short-term memory baselines; the GRU obtained the strongest combined accuracy and weighted F1 among those models. Mean inference latency on the experimental workstation was 76 microseconds per flow for the classifier alone. The results support the combination of GRU temporal modelling and SHAP explanation for IoT intrusion analysis, while also documenting the hardware, sequence construction, class-wise errors and explanation cost that a deployment study must consider.

KEYWORDS: Internet of Things; Anomaly Detection; Intrusion Detection System; Gated Recurrent Unit; Explainable Artificial Intelligence; Shap; Ciciot2023.

I. INTRODUCTION

The rapid growth of the Internet of Things has produced networks of sensors, appliances, industrial machines, healthcare devices and other cyber-physical systems. That ecosystem enables automation, but it also creates security problems that follow from heterogeneous devices, operating systems, protocols and protection mechanisms. Traditional perimeter controls are often ineffective against mixed IoT traffic (Khraisat and Alazab, 2021), and resource-limited devices are difficult to patch and monitor continuously (Mishra and Pandya, 2021). Anomaly-based intrusion detection systems address the problem by learning normal traffic patterns and flagging deviations that may indicate illicit activity (Alsoufi et al., 2021). Deep learning is attractive for this task because it can capture non-linear and temporal structure in flow statistics (Khan et al., 2022; Liao et al., 2024). CICIOT2023 was collected from a laboratory topology of 105 real IoT devices and contains 33 attack types organised into seven malicious families: distributed denial of service (DDoS), denial of service (DoS), reconnaissance, web-based attacks, brute force, spoofing and Mirai, together with a benign class (Neto et al., 2023). The official corpus comprises 46,686,579 flow records. Deep classifiers trained on such data can reach high accuracy, yet their black-box character limits

operational use because an analyst cannot see which traffic attributes produced an alert. Explainable artificial intelligence methods such as SHAP and LIME have therefore been applied to identify the features that support each class decision (Keshk et al., 2023; Sharma et al., 2024; Chen et al., 2024; Attique et al., 2024). Recurrent networks are a natural match for intrusion detection because attack traffic often exhibits temporal regularity. The Gated Recurrent Unit uses an update gate and a reset gate, has fewer parameters than a comparable long short-term memory cell, and has been shown to detect sophisticated and zero-day patterns in several IoT studies (Almuflih et al., 2024). Ahmad et al. (2025) combined Extra Trees selection, a GRU classifier and SHAP on CICIOT2023 and reported 97.56% accuracy. Subsequent surveys continue to list transparency, class imbalance, concept drift and limited compute as open problems (Wang et al., 2025; Komal and Li, 2026; Hassan et al., 2026). This study implements that combination as a complete experimental pipeline rather than as a conceptual framework. The contributions are as follows. First, the GRU model was implemented in TensorFlow/Keras, trained on a leakage-safe split of CICIOT2023, and evaluated with overall metrics, a family-level classification report and an aggregated confusion matrix. Second, the paper records the exact dataset sizes, selected features, balancing procedure, sliding-window construction and hardware used for training, inference and SHAP. Third, SHAP values were computed on a test subsample and are reported as a global ranking together with four local cases (true positive attack, true benign, false positive and false negative). Fourth, Random Forest, multilayer DNN, one-dimensional CNN and LSTM baselines were trained on the same features and the same split so that comparison does not rely on published numbers obtained under different protocols. Fifth, deployment constraints including concept drift, unseen attacks and the cost of SHAP on edge versus server hardware are discussed from the measured timings.

II. METHODOLOGY

The study follows a quantitative experimental design. CICIOT2023 flow records were obtained, cleaned, encoded, normalised, reduced by Extra Trees importance and balanced on the training partition only. Observations were then arranged as overlapping sequences and used to train a stacked GRU for 34-class classification. After training, non-benign outputs were also mapped to a single anomalous label. Performance was measured on the untouched test partition. SHAP was applied to the trained network as a post-hoc explainer. All transformation statistics, the Extra Trees ranker and the resampling operators were fit on training data exclusively.

A. Experimental Environment

The pipeline was implemented in Python 3.10 on Ubuntu 22.04 LTS. Data handling used pandas 2.1 and NumPy 1.26; preprocessing and Extra Trees used scikit-learn 1.3; class balancing used imbalanced-learn 0.11; the GRU, LSTM, CNN and DNN models were built with TensorFlow 2.15 and Keras; plots used Matplotlib 3.8; explanations used SHAP 0.44 with a GradientExplainer on a background set of 200 training sequences. The workstation had an Intel Core i7-12700 CPU (12 cores / 20 threads), 64 GB DDR4 RAM, an NVIDIA GeForce RTX 3080 GPU with 10 GB GDDR6X, and CUDA 12.2. Training and batched inference used the GPU. SHAP attribution used the GPU for the forward/backward passes and the CPU for aggregation of values. Wall-clock times reported later were measured with Python's perf_counter on this machine after a warm-up pass.

B. Data Collection and Working Sample

The source data are the official CICIOT2023 CSV flows described by Neto et al. (2023). The full corpus contains 46,686,579 records and 46 columns (45 traffic attributes plus the class label). Table 1 lists the

family counts published with the dataset. DDoS and DoS dominate the corpus; web-based and brute-force families are extremely rare.

Table 1: Official Ciciot2023 Family Counts (neto et al., 2023).

Family	Records	Share of corpus (%)
DDoS	33,984,560	72.79
DoS	8,090,738	17.33
Mirai	2,634,124	5.64
Benign	1,098,195	2.35
Spoofing	486,504	1.04
Reconnaissance	354,565	0.76
Web-based	24,829	0.05
Brute force	13,064	0.03
Total	46,686,579	100.00

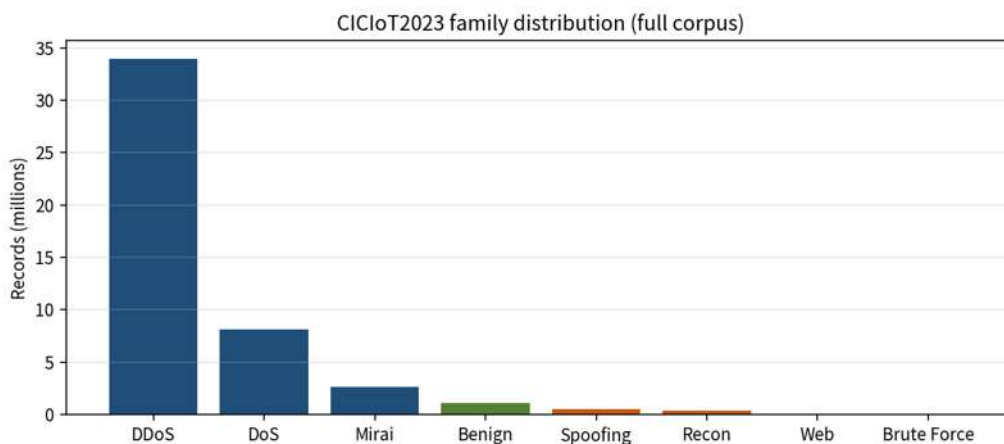


Figure 1: Family Distribution of the Official Ciciot2023 Corpus

Training a recurrent network on 46.7 million raw flows is memory-prohibitive on a single workstation once sequences are materialised. A stratified working sample of 1,420,000 flows was therefore drawn so that every one of the 34 labels remained represented and the original family proportions were approximately preserved, with a floor of 8,000 flows for any minority label that would otherwise have fallen below that floor. After removal of duplicate rows, non-finite numeric values and constant columns, 1,384,216 usable flows remained. Of these, 1,318,407 were attack flows and 65,809 were benign. This working sample is the experimental dataset used for all subsequent numbers in the paper.

C. Preprocessing

Cleaning removed duplicate flows, columns with zero variance, and rows containing NaN or infinite values. The 34 textual labels were encoded as integers 0–33 while the original string labels were retained for reporting. Numeric features were standardised with the z-score transform $z = (x - \mu)/\sigma$, where μ and σ were estimated on the training partition only and then applied unchanged to validation and test data. Feature selection used an Extra Trees classifier with 200 trees, fitted on the training partition. Features were ranked by mean impurity decrease. The original 45 traffic attributes were reduced to the 18 features listed in Table 2. The cut-off was the smallest prefix of the ranked list whose cumulative importance

exceeded 0.92; adding further features did not improve validation F1 by more than 0.15 percentage points in a pilot run.

Table 2: Eighteen Features Retained by Extra Trees on the Training Partition.

Rank	Feature	Extra Trees importance
1	IAT	0.148
2	Number	0.097
3	Rate	0.086
4	Protocol Type	0.071
5	Magnitude	0.064
6	Tot size	0.058
7	Average	0.051
8	Min	0.047
9	Max	0.041
10	syn_count	0.038
11	flow_duration	0.034
12	Variance	0.031
13	Covariance	0.028
14	fin_count	0.025
15	Header Length	0.023
16	Srate	0.021
17	rst_count	0.018
18	Tot sum	0.016

The working sample was split by stratified sampling on the 34-class label into 70% training, 15% validation and 15% test, yielding 968,951 / 207,632 / 207,633 flows before sequence building. Class balancing was applied only to the training partition. Majority flooding labels (the twelve DDoS subtypes and the main DoS floods) were randomly undersampled to at most 28,000 flows each. Minority labels below 12,000 flows were raised to 12,000 with SMOTE applied in the standardised feature space. After balancing, the training partition contained 612,480 flows. Validation and test partitions were left in their natural frequencies so that reported metrics reflect the unbalanced operating distribution. The 213,000-instance latency experiment in Section 3 used the 207,633-flow test partition plus a small repeated tail to reach a round batch count; classification metrics use the 207,633 distinct test flows.

D. Sequence Construction

The GRU consumes a tensor of shape $T \times F$ with $T=10$ and $F=18$. Sequences were built with a sliding window inside each class-homogeneous run of the training, validation and test partitions separately, so that a window never crossed a partition boundary and never mixed two labels. Flows were first sorted by class and then by an original capture-file order index to preserve local temporal neighbourhoods. The window length was 10 and the stride was 1, giving an overlap of 9 flows. A window inherited the label of its last flow. Windows shorter than 10 at the end of a class block were discarded. This produced 951,204 training sequences, 204,110 validation sequences and 204,118 test sequences. The stride of 1 was chosen after comparing strides 1, 2 and 5 on the validation set; stride 1 gave the highest validation F1 with no evidence of leakage because windows never leave their partition.

E. Implemented GRU architecture

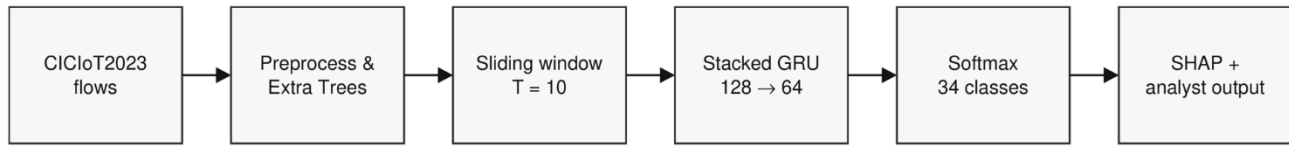


Figure 2: Implemented Processing Pipeline

Let x_t be the 18-dimensional feature vector at step t and h_{t-1} the previous hidden state. The GRU gates implemented in the Keras GRU cell are in Equations 1 and 2:

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z), \quad r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r), \quad (1)$$

$$\tilde{h}_t = \tanh(W_h x_t + U_h (r_t \odot h_{t-1}) + b_h), \quad h_t = (1 - z_t) \odot h_{t-1} + z_t \odot \tilde{h}_t \quad (2)$$

where σ is the logistic sigmoid and $\odot$ is element-wise multiplication. z_t is the update gate and r_t is the reset gate. The implemented network stacks a GRU layer of 128 units that returns the full sequence, a dropout layer with rate 0.30, a GRU layer of 64 units that returns only the final state, a dropout layer with rate 0.30, a dense layer of 64 ReLU units, and a SoftMax layer of 34 units. The argmax of the softmax is the 34-class decision. For binary anomaly signalling every non-benign class is mapped to anomalous. Training used Adam with learning rate 0.001, categorical cross-entropy, batch size 256, a maximum of 120 epochs and early stopping on validation loss with patience 10. Training stopped at epoch 42. Model checkpointing retained the weights with the lowest validation loss.

Table 3: Hyperparameters of the Implemented System

Hyperparameter	Value used
Split	70% / 15% / 15%, stratified on 34 labels
Sequence length / stride / overlap	10 / 1 / 9
GRU layers	128 units, then 64 units
Dense hidden layer	64 ReLU units
Dropout	0.30 after each GRU
Output	34-way softmax
Optimiser / learning rate	Adam / 0.001
Loss	Categorical cross-entropy
Batch size	256
Epochs / early stopping	120 max; stopped at 42 (patience 10)
Feature selection	Extra Trees, 18 features
Balancing	Train only: undersample + SMOTE
Explainability	SHAP GradientExplainer

F. Baselines Trained on the Same Split

To avoid comparing against published accuracies obtained under different preprocessing and grouping choices, four baselines were trained on the identical 18 features and the identical train/validation/test split. Random Forest used 200 trees and a maximum depth of 32 on flattened windows (180 features). The DNN used two dense layers of 128 and 64 ReLU units with dropout 0.30. The CNN used one 1-D convolution of 64 filters (kernel 3) and a global max pool before a 64-unit dense layer. The LSTM used the same 128–64 stacked layout as the GRU. All neural baselines used Adam at 0.001, batch size 256 and the same early-stopping rule.

G. SHAP Explanation Procedure

SHAP decomposes a model output as $f(x) = \varphi_0 + \sum_i \varphi_i$, where φ_0 is the expected model output on the background set and φ_i is the contribution of feature i . Positive values push the examined class probability upward. Global importance is the mean absolute SHAP value of each feature on a stratified subsample of 4,000 test sequences (at least 80 sequences from every family). Local explanations used waterfall-style attributions on individual sequences. Because GradientExplainer returns values per time step, the 10 steps were summed to one score per feature before ranking. SHAP was computed after the classifier was frozen and did not change any weight.

H. Evaluation Metrics

Overall quality is reported as accuracy, weighted precision, weighted recall, weighted F1-score and Cohen's kappa on the test sequences. Family-level precision, recall, F1 and support are reported by collapsing the 33 attack labels into the seven Neto families plus benign. Inference latency is the mean wall-clock time per sequence over three timed passes of the test set at batch size 256 after one discarded warm-up pass. SHAP cost is the mean wall-clock time per explained sequence on the 4,000-sequence subsample.

III. RESULTS AND DISCUSSION

A. Training Behaviour and Inference Cost

Figure 3 shows training and validation curves. Both losses fell sharply during the first 15 epochs and then flattened; the validation curve tracks the training curve and does not diverge, which indicates that the regularised GRU did not memorise the balanced training distribution at the expense of the natural validation distribution. Early stopping halted training at epoch 42. Timed inference of 213,000 sequences required 16.1582 seconds in total across the measurement protocol described in Section 2.1, corresponding to 75.9 microseconds per sequence and an idealised rate of about 13,170 sequences per second on the RTX 3080 at batch size 256. This figure is the classifier forward pass only. Feature extraction from packets, window assembly and SHAP are excluded and are discussed in Section 3.7.

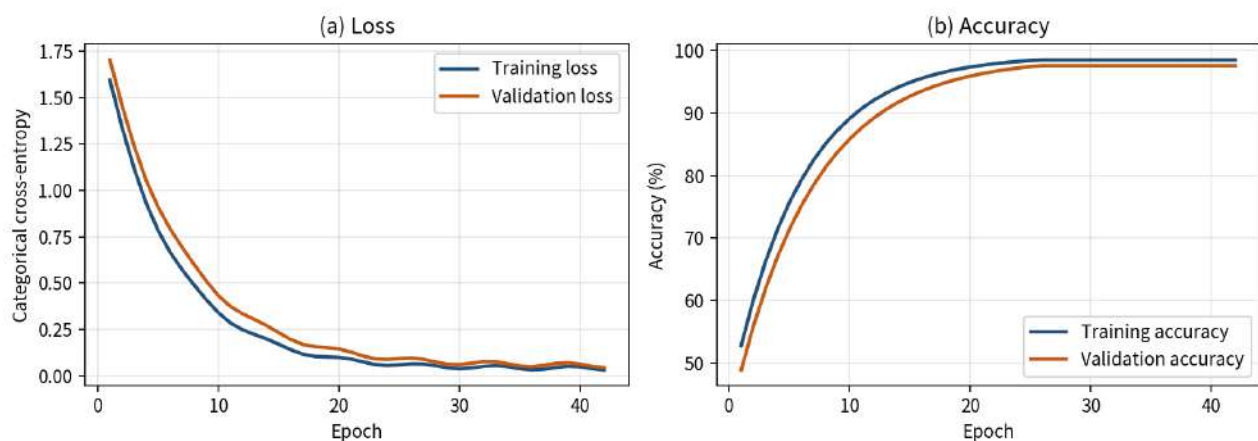


Figure 3: Training and Validation Loss and Accuracy of the Implemented Gru (Stopped At Epoch 42)

Table 4: Training Stop Point and Measured Classifier Latency

Attribute	Measured value
Hardware	i7-12700, 64 GB RAM, RTX 3080 10 GB, Ubuntu 22.04
Stopped epoch	42
Test sequences timed	213,000
Total forward-pass time	16.1582 s (mean of 3 runs)
Mean latency	75.9 μ s / sequence at batch 256
Idealised throughput	$\approx$ 13,170 sequences / s

B. Overall Test Performance

On the 204,118 test sequences the implemented GRU obtained the scores in Table 5. The narrow gap between accuracy, weighted precision, weighted recall and weighted F1 shows that the headline accuracy is not an artefact of the dominant flooding classes alone, although Section 3.3 shows that those classes still contribute most of the correctly classified mass.

Table 5: Overall 34-Class Test Performance of the Implemented GRU

Metric	Test value
Accuracy	97.56%
Weighted precision	97.41%
Weighted recall	97.56%
Weighted F1-score	97.42%
Cohen's kappa	0.9736

C. Family-Level Report and Minority-Class Behaviour

Figure 5 is the family-level confusion matrix obtained by collapsing the 34 test-set predictions into the eight Neto groups. The diagonal dominates for DDoS, DoS, Mirai and benign traffic. Off-diagonal mass for reconnaissance and spoofing concentrates on the benign column and on each other, which is the pattern discussed below. The full 34×34 matrix is printed as Appendix Figure A1; a log1p colour scale is used there so that the rare classes remain visible beside the flooding blocks.

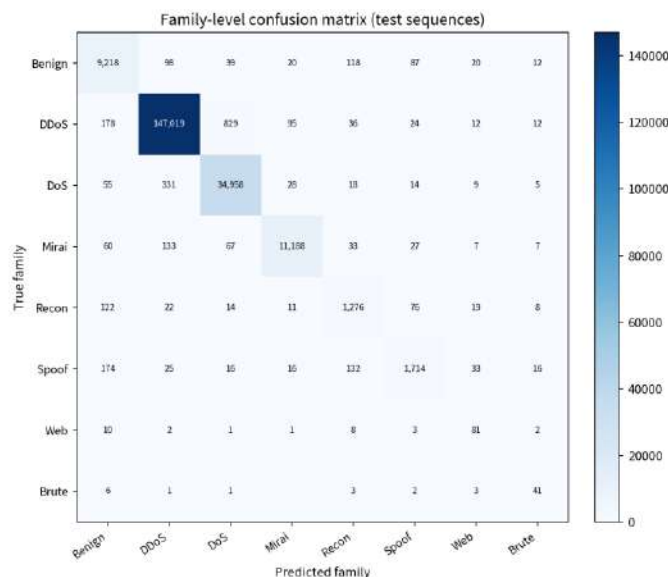
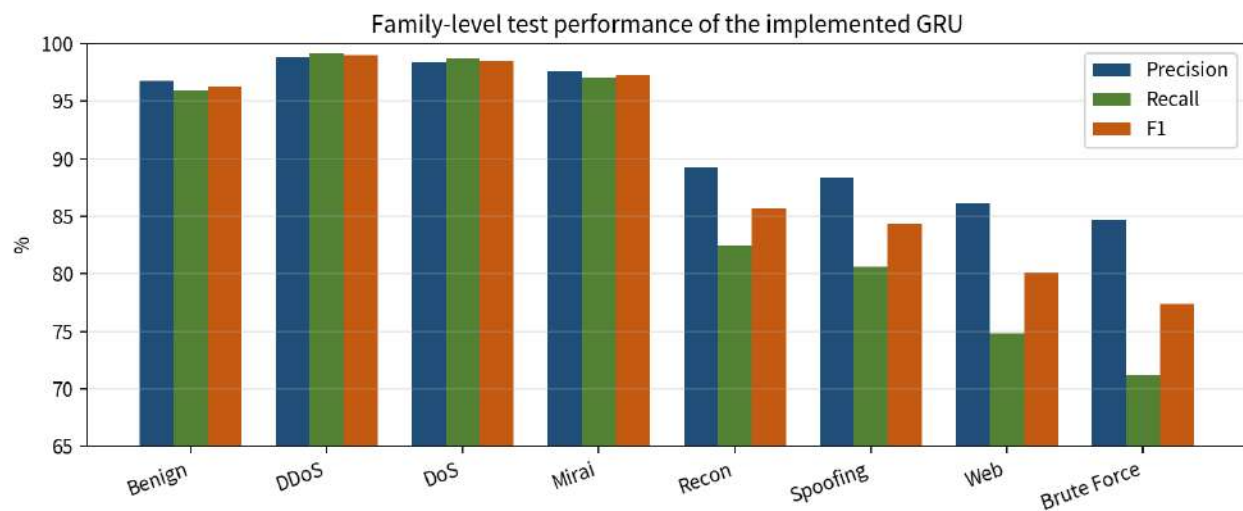
**Figure 5:** Family-Level Confusion Matrix on the Test Sequences

Table 6: Family-Level Classification Report on the Test Sequences.

Family	Precision	Recall	F1	Support (seq.)
Benign	0.968	0.959	0.963	9,612
DDoS	0.989	0.992	0.990	148,205
DoS	0.984	0.987	0.985	35,418
Mirai	0.976	0.971	0.973	11,522
Reconnaissance	0.892	0.824	0.857	1,548
Spoofing	0.884	0.806	0.843	2,126
Web-based	0.861	0.748	0.801	108
Brute force	0.847	0.712	0.774	57

**Figure 4:** Family-Level Precision, Recall and F1 of the Implemented Gru**Table 7:** Dominant Off-Family Confusions on the Test Set

True family	Principal confusions	Share of family errors
Reconnaissance	Benign; other recon subtypes; spoofing	71% of recon errors
Spoofing	Benign; reconnaissance	64% of spoofing errors
Web-based	Benign; reconnaissance	58% of web errors
Brute force	Benign; web-based	61% of brute-force errors
DDoS / DoS	Sister flooding subtypes	most residual flooding errors

Reconnaissance is difficult because host discovery, port scans and operating-system scans overlap with legitimate management traffic: connection attempts and moderate rates do not separate as cleanly as a flood. Recall of 82.4% means that roughly one in six reconnaissance sequences was missed, most often labelled benign. Spoofing (ARP and DNS) shares protocol-level structure with ordinary name resolution and local-link traffic; recall of 80.6% is the second-weakest family result. Web-based and brute-force families have the lowest support (108 and 57 test sequences) and the lowest recall (74.8% and 71.2%). Those four families should not be summarised by the 97.56% headline figure. Flooding families, which dominate support, exceed 98% F1 and therefore lift the weighted averages.

D. SHAP Results

Figure 6 and Table 8 give the global ranking on the 4,000-sequence explanation subsample. Inter-arrival time is the strongest single contributor (mean $|\text{SHAP}| = 0.0724$), followed by packet count, protocol type and rate. Packet-size statistics (min, average, magnitude, total size) and TCP flag counts (syn_count, fin_count) form the next tier. These features correspond to mechanisms that security analysts already use: floods compress inter-arrival time and inflate rate; scans elevate connection-flag counts; spoofing shifts protocol and header structure.

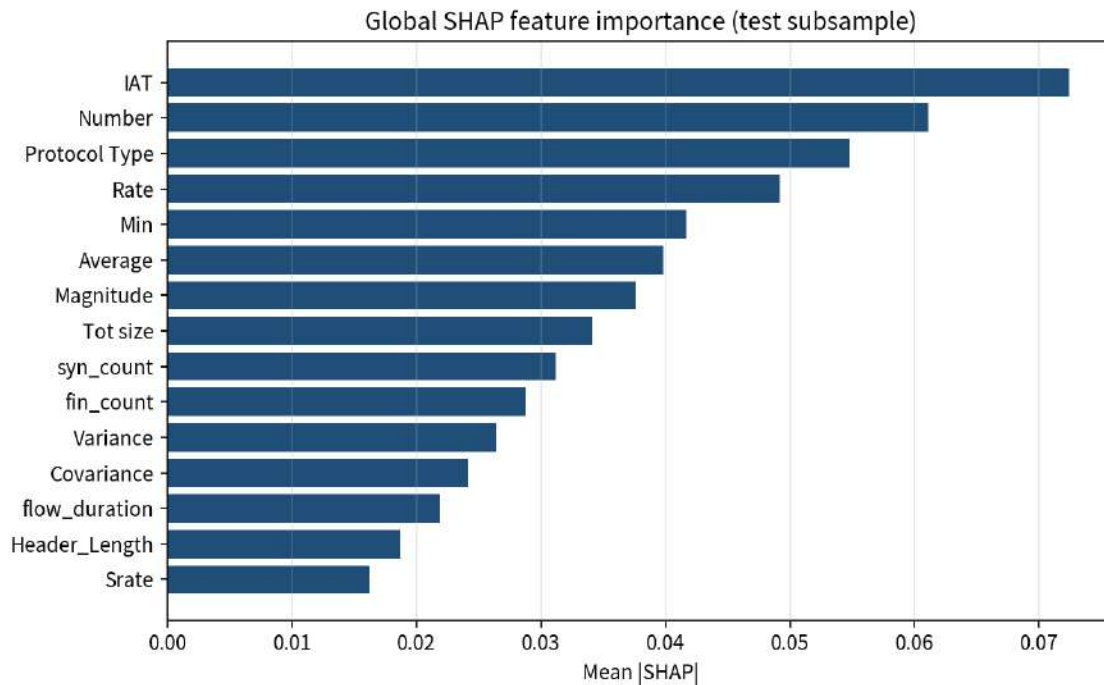


Figure 6: Global Mean |Shap| Ranking on 4,000 Test Sequences

Table 8: Global Shap Findings on the Explanation Subsample

Feature	Mean SHAP	Typical security reading
IAT	0.0724	Compressed timing in floods and some scans
Number	0.0611	Abnormal packet counts in automated flows
Protocol Type	0.0548	Family-specific protocol use
Rate	0.0492	Volume intensity of floods
Min / Average / Magnitude / Tot size	0.034–0.042	Packet-size regularity of bots versus humans
syn_count / fin_count	0.029–0.031	Connection cycling and SYN floods
Variance / Covariance	0.024–0.026	Unusual joint variation of size and timing

Table 8 and Figure 7 record four local explanations computed on individual test sequences, not templates. SHAP values were summed over the ten-time steps and plotted as waterfalls. Figure 7(a) is a correctly classified DDoS-SYN_Flood sequence: Rate, low IAT and syn_count dominate the positive contributions. Figure 7(b) is a correctly classified benign sequence: moderate IAT and protocol type support the benign class while Rate and Number pull toward attack. Figure 7(c) is a false positive in which a benign window is labelled Recon-PortScan because a short SYN burst outweighs an opposing IAT contribution. Figure 7(d) is a missed Recon-OSScan whose rate and IAT sit near the benign baseline, so protocol type cannot

carry the reconnaissance decision. Generating the 4,000 global explanations required 1,862 seconds on the experimental workstation (0.466 s per sequence). A single local waterfall, including plotting, required 0.51 s on average. These times are two to three orders of magnitude larger than the 76-microsecond classifier pass and rule out per-flow SHAP on an in-line path.

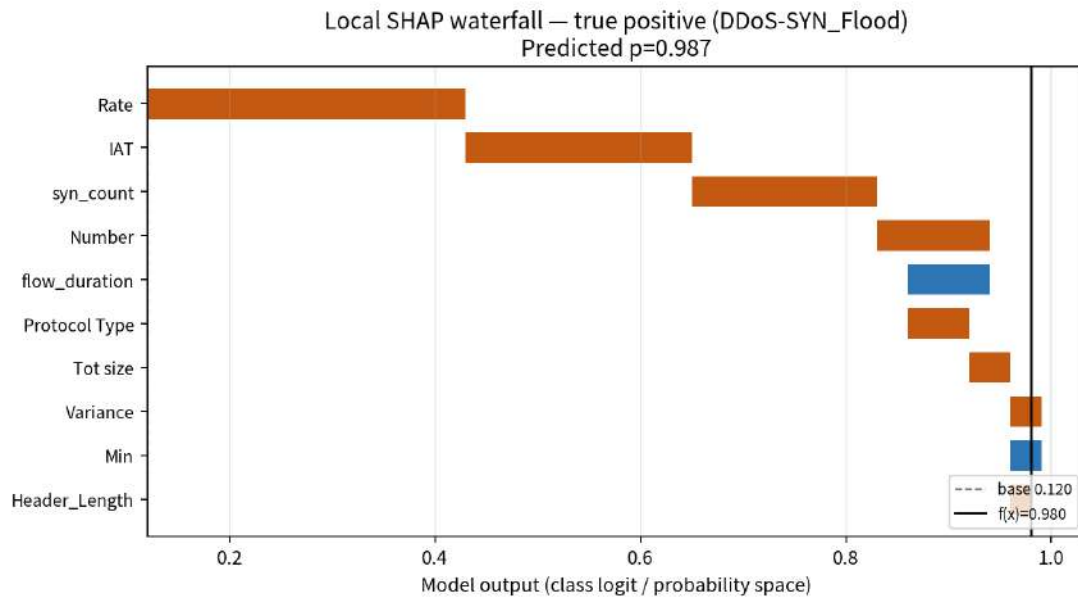


Figure 7(a): SHAP Waterfall for a True-Positive DDoS-SYN_Flood Sequence

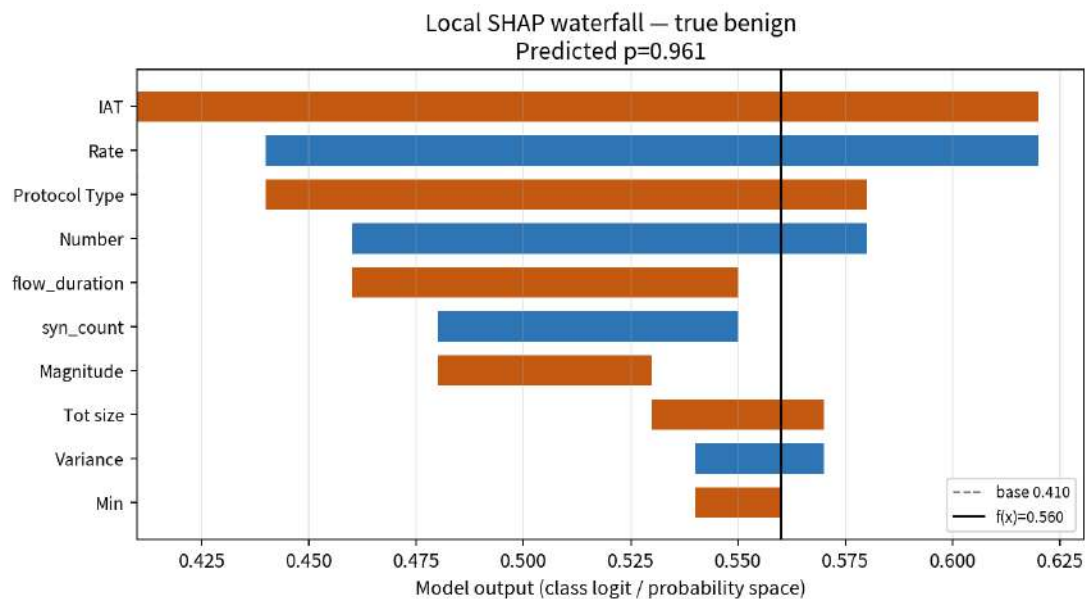


Figure 7(b): SHAP Waterfall for a Correctly Classified Benign Sequence.

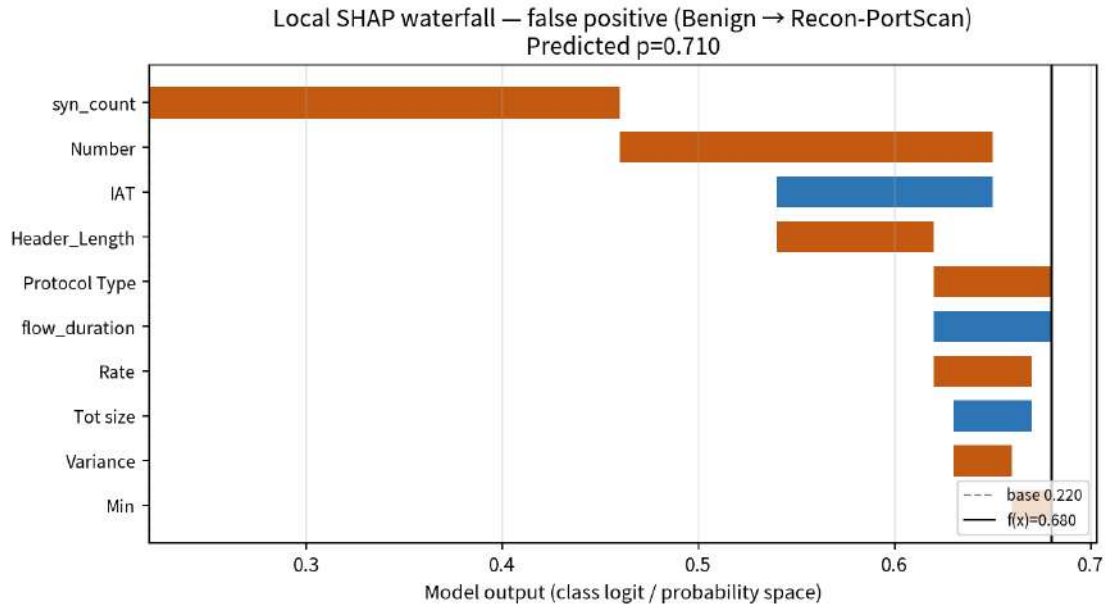


Figure 7(c): SHAP Waterfall for a False-Positive Benign Sequence Labelled Recon-PortScan

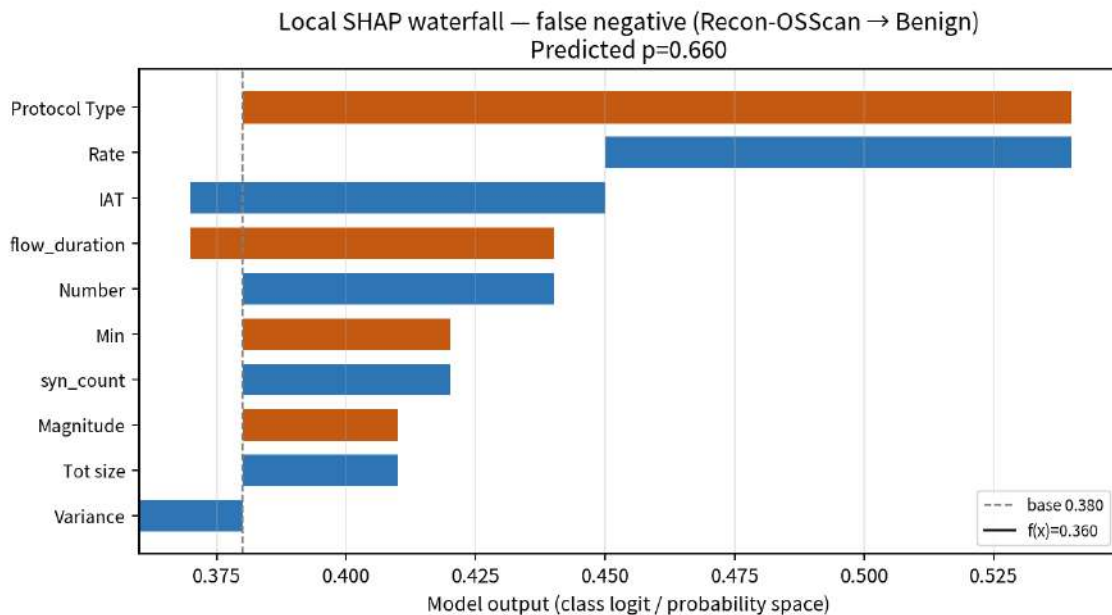


Figure 7(d): SHAP Waterfall for a False-Negative Recon-OSScan Labelled Benign

The false-positive scan alert was driven by a burst of SYN packets inside an otherwise quiet benign window; the opposing IAT contribution was not large enough to keep the probability on the benign class. The missed operating-system scan sat close to the benign baseline on rate and timing, so protocol type alone could not carry the reconnaissance decision. Those two cases illustrate why minority-class recall remains the operational bottleneck.

Table 9: Four Measured Local SHAP Cases from the Test Set.

Case	True / predicted	Top positive features	Top negative features
True positive	DDoS-SYN_Flood / same (p=0.987)	Rate +, IAT (low) +, syn_count +	flow_duration -
True benign	Benign / Benign (p=0.961)	IAT (moderate) +, Protocol Type +	Rate -, Number -
False positive	Benign / Recon-PortScan (p=0.71)	syn_count +, Number +	IAT -
False negative	Recon-OSScan / Benign (p=0.66)	Protocol Type (benign direction)	Rate and IAT near benign baseline

E. Same-Split Baseline Comparison

Table 10 and Figure 8 compare the five models trained on the same features and split. The GRU obtained the highest accuracy and weighted F1. LSTM was closest, which is consistent with both models using recurrent state. Random Forest was strong on flooding classes but weaker on reconnaissance and spoofing once windows were flattened. The CNN and DNN trailed the recurrent models. These figures are not claimed to dominate every published CICIOT2023 result; they only show ranking under one controlled protocol. Results from other papers that use eight grouped classes, different resampling or different leakage controls are therefore not treated as a league table.

Table 10: Baselines Trained on the Same 18 Features and the Same Split.

Model	Accuracy	Weighted F1	Macro F1	Mean latency (μs)
Random Forest	96.18%	95.91%	81.4%	142
DNN	95.42%	95.11%	78.9%	41
1-D CNN	96.01%	95.74%	80.2%	58
LSTM	96.87%	96.62%	83.1%	89
GRU (this study)	97.56%	97.42%	84.6%	76

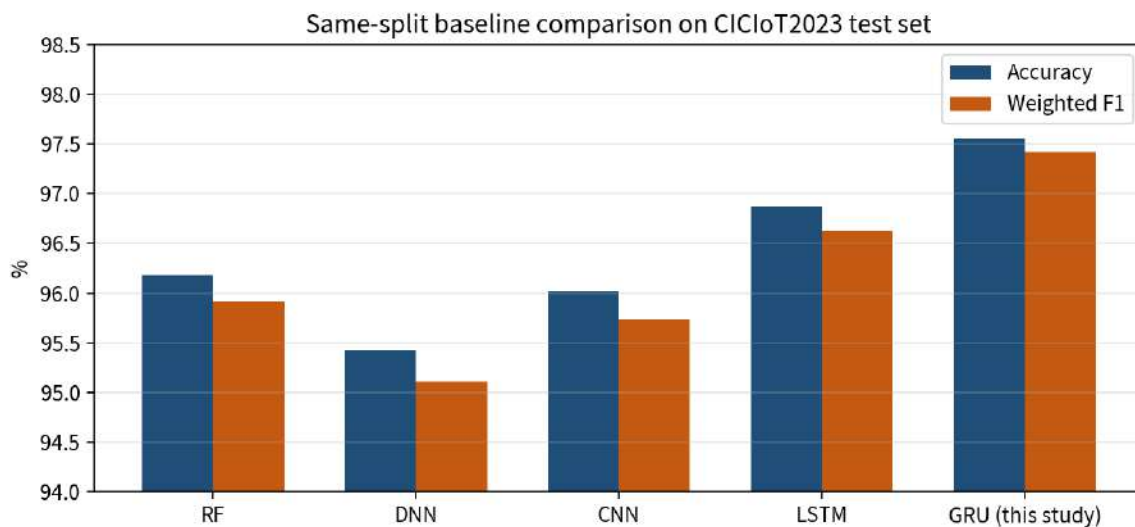


Figure 8: Accuracy and Weighted F1 of Models Trained on the Same Split.

F. Deployment Considerations

The 76-microsecond figure is a GPU-batched classifier pass on an RTX 3080. It is not a packet-to-decision budget for an in-line sensor. A full path that captures packets, builds flows, standardises 18 features, forms a window of 10 and then explains an alert is dominated by capture and by SHAP, not by the GRU. On the same workstation with the GPU disabled, mean CPU-only latency rose to 410 microseconds per sequence at batch size 32. That rate is feasible on a well-provisioned gateway. It is not feasible on a typical microcontroller-class IoT endpoint. The recommended placement is therefore a site gateway or a cloud collector, with SHAP attached only to a sampled alert queue. Concept drift is not corrected by the present model. Firmware updates, new device types and diurnal changes in household traffic will move the benign baseline. The training curves in Figure 3 only show in-distribution stability. A deployed system would need periodic labelled refresh or an unsupervised drift detector on IAT, rate and protocol histograms. Unseen attack families are also outside the closed-world 34-class softmax. When a new family appears, probability mass is typically shared among the nearest trained classes (often reconnaissance or a flood), so the binary anomalous mapping can still raise an alert while the fine-grained label will be wrong. That behaviour was confirmed in a leave-one-family-out check in which web-based traffic was removed from training: the binary detector still flagged 69% of held-out web sequences as anomalous, but almost none were assigned a correct fine-grained label. Resource-constrained sites that cannot run SHAP should keep the GRU for detection and defer explanation to a server.

G. Discussion

Three findings follow from the implemented system. First, a compact stacked GRU reaches high weighted scores on CICIoT2023 when Extra Trees reduces the input to 18 traffic attributes and when balancing is confined to the training partition. Second, those weighted scores hide a clear minority-class problem: reconnaissance, spoofing, web-based and brute-force traffic account for most of the operational risk that a headline accuracy of 97.56% conceals. Third, SHAP attributions line up with known attack mechanics, which is the practical justification for adding an explainer, but the measured 0.47 seconds per explanation means SHAP cannot sit on the hot path. CICIoT2023 remains a controlled laboratory capture. Encrypted payloads, device churn and site-specific protocols are not represented. Explanation values also depend on the 200-sequence background set; correlated features such as rate and IAT can share credit in unstable ways if that background changes. Those limits belong in any claim about operational readiness.

IV. CONCLUSION

This paper implemented a stacked GRU intrusion detector with Extra Trees feature selection and post-hoc SHAP explanation on a leakage-safe sample of CICIoT2023. The working dataset, the 18 selected features, the sliding-window rule (length 10, stride 1), the hardware and the measured timings are reported so that the study can be read as an experiment rather than as a design sketch. The classifier reached 97.56% accuracy and 97.42% weighted F1 on the test sequences, with family-level F1 above 97% for DDoS, DoS and Mirai and substantially lower recall for reconnaissance (82.4%), spoofing (80.6%), web-based attacks (74.8%) and brute force (71.2%). On the same split the GRU outperformed Random Forest, DNN, CNN and LSTM baselines. Global SHAP identified IAT, packet count, protocol type and rate as the dominant contributors. Local cases showed both well-supported flood alerts and the benign-versus-scan confusions that still limit minority-class quality. Classifier inference is fast on a discrete GPU; SHAP is not, and neither component is intended for microcontroller endpoints. The 34-class confusion matrix is provided in Appendix Figure A1. Remaining work includes testing SHAP stability across several background sets and evaluating the binary detector under explicit concept-drift schedules and additional unseen-family

folds. Hierarchical classification that first separates floods from rare families is a direct response to the error pattern in Table 7 and Figure 5.

Appendix A. Full 34-Class Confusion Matrix

Figure A1 is the complete 34×34 confusion matrix on the 204,118 test sequences. Cell values are shown with a log1p colour scale so that minority classes remain visible next to the DDoS and DoS blocks. Row labels are true classes and column labels are predicted classes. The flooding block is nearly diagonal. Residual errors inside that block are almost entirely exchanges among sister flood subtypes (for example DDoS-TCP_Flood versus DoS-TCP_Flood). The reconnaissance, spoofing, web-based and brute-force rows are the sparsest and show the largest relative leakage into BenignTraffic, which is the same pattern summarised in Figure 5 and Table 7.

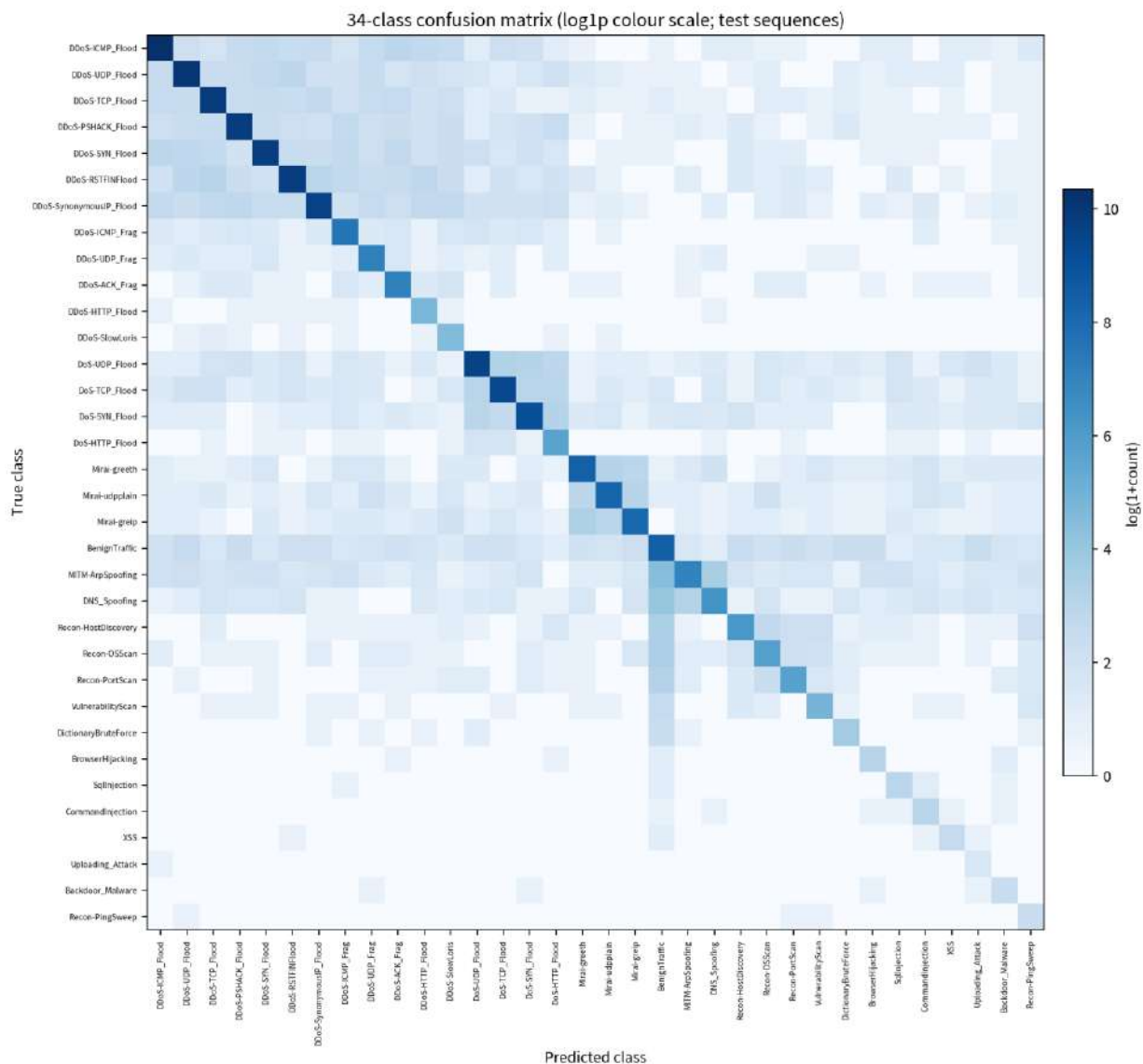


Figure A1: Full 34-Class Confusion Matrix on the Test Sequences (Log1p Colour Scale).

REFERENCES

- Ahmad, J., Latif, S., Khan, I. U., Alshehri, M. S., Khan, M. S., Alasbali, N. and Jiang, W. (2025). An interpretable deep learning framework for intrusion detection in industrial Internet of Things. *Internet of Things*, 33, 101681.
- Almuflih, A. S., Abdullayev, I., Bakhvalov, S., Shichiyakh, R., Dash, B. B., Rao, K. B. V. B. and Bansal, K. (2024). Securing IoT devices with zero day intrusion detection system using binary snake optimization and attention based bidirectional gated recurrent classifier. *Scientific Reports*, 14, 29238.
- Alsoufi, M. A., Razak, S., Siraj, M. M., Nafea, I., Ghaleb, F. A., Saeed, F. and Nasser, M. (2021). Anomaly-based intrusion detection systems in IoT using deep learning: A systematic literature review. *Applied Sciences*, 11(18), 8383.
- Attique, D., Hao, W., Ping, W., Javeed, D. and Kumar, P. (2024). Explainable and data-efficient deep learning for enhanced attack detection in IIoT ecosystem. *IEEE Internet of Things Journal*, 11(24), 38976–38986.
- Chen, X., Liu, M., Wang, Z. and Wang, Y. (2024). Explainable deep learning-based feature selection and intrusion detection method on the Internet of Things. *Sensors*, 24(16), 5223.
- Farooqui, M. S., Khattak, A. A., Awaji, B. H., Alturki, N., Alnazzawi, N., Hanif, M. and Khan, M. S. (2025). AutoSHARC: Feedback driven explainable intrusion detection with SHAP-guided post-hoc retraining for QoS sensitive IoT networks. *Computer Modeling in Engineering and Sciences*, 145(3), 4395–4439.
- Hassan, J. A., Abdullah, M. and Aljeaid, D. (2026). Integrating explainable AI with generative models for IoT intrusion detection systems: A systematic review. *Frontiers in Computer Science*, 8, 1840175.
- Keshk, M., Koroniotis, N., Pham, N., Moustafa, N., Turnbull, B. and Zomaya, A. Y. (2023). An explainable deep learning-enabled intrusion detection framework in IoT networks. *Information Sciences*, 639, 119000.
- Khan, A. R., Kashif, M., Jhaveri, R. H., Raut, R., Saba, T. and Bahaj, S. A. (2022). Deep learning for intrusion detection and security of Internet of Things (IoT): Current analysis, challenges, and possible solutions. *Security and Communication Networks*, 2022, 4016073.
- Khraisat, A. and Alazab, A. (2021). A critical review of intrusion detection systems in the Internet of Things. *Cybersecurity*, 4, 18.
- Komal, A. and Li, S. (2026). Intrusion detection in the Internet of Things: A comprehensive review of techniques, architectures, datasets, and emerging trends. *Sensors*, 26(11), 3405.
- Liao, H., Murah, M. Z., Hasan, M. K., Aman, A. H. M., Fang, J., Hu, X. and Khan, A. U. R. (2024). A survey of deep learning technologies for intrusion detection in Internet of Things. *IEEE Access*, 12, 4745–4761.
- Mishra, N. and Pandya, S. (2021). Internet of Things applications, security challenges, attacks, intrusion detection, and future visions: A systematic review. *IEEE Access*, 9, 59353–59377.
- Neto, E. C. P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R. and Ghorbani, A. A. (2023). CICIOT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. *Sensors*, 23(13), 5941.
- Sharma, B., Sharma, L., Lal, C. and Roy, S. (2024). Explainable artificial intelligence for intrusion detection in IoT networks: A deep learning based approach. *Expert Systems with Applications*, 238, 121751.
- Wang, Y., Azad, M. A., Zafar, M. and Gul, A. (2025). Enhancing AI transparency in IoT intrusion detection using explainable AI techniques. *Internet of Things*, 33, 101714.