International Journal of Scientific Research and Modern Technology (IJSRMT) ijsrmt.com
Volume 5, Issue 10, 2026
DOI: https://doi.org/10.38124/ijsrmt.v5i10.1775

Development of a Gated Recurrent Unit (GRU) Classifier for Detection of Malicious Network Flows

Kenechukwu Martin Agbo[1]; Chuka Charles Agubosim[2]

[1]Dave Umahi Federal University of Health Sciences, Uburu
[2]Department of Computer Science, Chukwuemeka Odumegwu Ojukwu University, Uli

Publication Date: 2026/11/13

Abstract

The Internet of Things (IoT) brings heterogenous network traffic which is sourced from resource-constrained devices. Detection of network intrusions can help provide an important functionality at the gateway. In this study, a Gated Recurrent Unit (GRU) model is developed to detect network flows containing malicious traffic using the RT-IoT2022 dataset. The dataset contained 123,117 records from 3 classes of benign data and 9 attack classes. After dropping the identifying fields, port numbers, opposing predictor variables and ensuring only one sample was available per remaining predictor variable, there were 18,265 samples. All numeric variables were log transformed and scaled using the training set. The protocol and service variables were converted using one-hot encoding so there were a total of 92 variables. A GRU model with 32 gates was tested with a single network flow. It was compared to logistic regression, linear support vector machine and random forest, all applied to the same split. The benchmark GRU model provided an accuracy, precision, recall, F1 score and AUC of 98.58%, 97.59%, 98.31%, 0.9795 and 0.9994 respectively using 2,740 samples in the test set. There were 23 false positives and 16 missed attacks. The mean accuracy across 3 seeds for the GRU was 98.60%. The random forest model had the highest test accuracy (99.42%). This paper provides a repeatable baseline for using GRUs on network flows. It demonstrates how important it is to consider false positives, duplicated samples and competitive classical models.

Keywords: *Internet of Things; Intrusion Detection; Gated Recurrent Unit; RT-IoT2022; Network Security; Flow Classification.*

I. INTRODUCTION

The Internet of Things (IoT) provides a network of connected sensors, smart lighting, devices and monitoring systems. This generates a variety of traffic patterns, which can cause issues regarding security and confidentiality if there was a security breach. An intrusion detection system (IDS) analyses traffic patterns and identifies any anomalies that may have occurred due to unauthorised access, scanning, denial of service and manipulation. Placing the intrusion detection system at the gateway reduces complex learning models being used on each end device (Gyamfi & Jurcut, 2022).

With the use of artificial intelligence, correlations between traffic samples and attacks can be learnt. However, complex models do not necessarily provide good results. Concept drift, high dimensionality and computational complexity are ongoing problems (Adnan et al., 2021). Multi layer intrusion detection and attack identification have been looked into within this paper (Qaddoura et al., 2021), showing the issue of distinguishing between binary filtering and detailed attack identification. Deep learning has also been investigated for intrusion detection in MQTT enabled environments (Khan et al., 2021) and shows how the traffic protocol and degree of aggregation affects the choice of feature.

GRU networks use gates to manage the hidden state. Sequential models such as GRU have been investigated for IoT servers (Zhong et al., 2021). GRU networks have a nice structure if a simplified representation is required, however the recurrent nature must match the data. The public dataset RT-IoT2022 does not have an explicit date column, just flow statistics. This means a GRU model for one flow has been investigated with the hidden state reset after each observation. The metrics provided do not show

Agbo, K. M., & Agubosim , C. C. (2026). Development of a Gated Recurrent Unit (GRU) Classifier for Detection of Malicious Network Flows. *International Journal of Scientific Research and Modern Technology*, *5*(10), 131–138. https://doi.org/10.38124/ijsrmt.v5i10.1775

progression of an attack, rather just how the flow should be classified.

Two contributions have been made. Firstly, a well defined GRU based binary classifier has been developed and applied to a public IoT benchmark. This includes an audit of the predictor signature prior to splitting. Secondly, the model has been compared with three classical models using the same features and out of sample records. Detection scores along with false positive and negative counts have been provided.

➢ *Related Studies*

Several studies have investigated sequential intrusion detection methods using network and application features (Zhong et al., 2021). Multi-layer approaches have been investigated which can detect and identify the type of attack (Qaddoura et al., 2021). Others have investigated deep learning techniques with MQTT data representation (Khan et al., 2021). This has motivated the use of representations but also shows different definitions of what constitutes a feature can be used depending on if the task is packet, flow or sequence based. This experiment will use a single full vector of features.

Edge computing, learning strategies and datasets have been reviewed (Gyamfi & Jurcut, 2022). The study 'Intrusion Detection in IoT Using Deep Learning' investigates different CNN, GRU and LSTM models giving motivation to using recurrent gates for comparison within IoT classification (Banaamah & Ahmad, 2022). New datasets have been introduced including the Edge-IIoTset generated from an IoT and Industrial IoT test bed (Ferrag et al., 2022) and CICIoT2023 for large scale attack benchmarking (Neto et al., 2023). This gives more options for future validation. However, RT-IoT2022 was chosen due to its small sized flow table which can be easily accessed.

The quantized autoencoder was investigated using RT-IoT2022 (Sharmila & Nagapadma, 2023a). A comparison of classical and neural classifiers was performed on this family of datasets (Airlangga, 2024). This provided motivation to use a tree based classifier as a competitive baseline. Complex gated recurrent networks were investigated (El-Shafeiy et al., 2024) and a hybrid CNN-LSTM-GRU was investigated for IoT electric vehicle charging (Kilichev et al., 2024). This experiment only uses a single gated classifier.

Hybrid CNN-GRU based models were investigated for IoT intrusion detection (Adefemi et al., 2025). Emphasis was placed on representation learning and dealing with imbalanced data. Different deep learning models were compared across different IoT datasets and loss functions (Waqas et al., 2025). This provided motivation to clearly state what benchmark was used, how many samples were used in the cleaned dataset, what seed was used for splitting and initialisation.

II. METHODOLOGY

Figure 1 presents the pipeline: binary labelling, predictor-signature auditing, partitioning, training-only preprocessing, classification and held-out evaluation. Validation loss selects the GRU checkpoint. A prespecified threshold of 0.5 provides the primary detection outcomes.

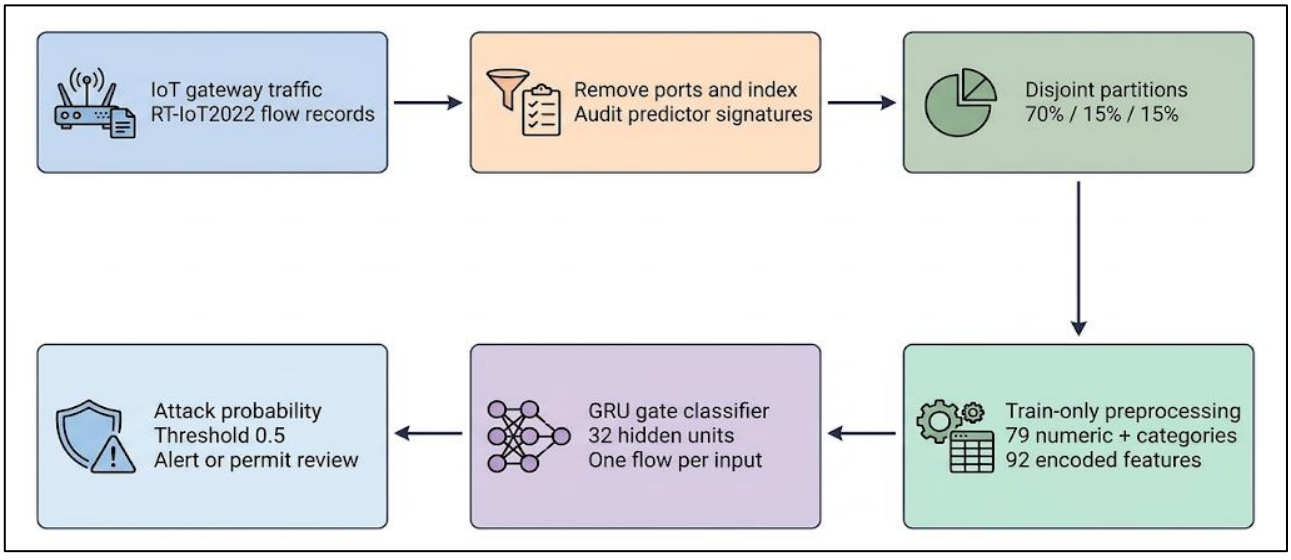


Fig 1 Experimental Pipeline for GRU-based IoT Flow Intrusion Detection.

➢ *Data Collection*

The RT-IoT2022 dataset has been taken from the UCI Machine Learning Repository (Sharmila & Nagapadma, 2023b). This dataset has been created using a number of IoT based infrastructures including ThingSpeak and a Wipro bulb and MQTT based devices. In terms of network monitoring, Zeek was used alongside a Flowmeter plugin. The dataset has been created using a number of attacks including SYN flood, Slowloris, ARP spoofing, SSH brute force and many types of Nmap scans. The data has been gathered using devices producing normal traffic, however, attacks have been artificially added to the data. The dataset consists of 123,117 rows after downloading the data. During export, the row index has also been recorded, along with 83 predictors and the target variable 'Attack_type'.

The target variable in this case is binary, being equal to y = 0 for Thing_Speak, MQTT_Publish and Wipro_bulb, and y = 1 for all other attacks. Table 1 shows the frequencies for each label. These labels have come directly from the downloaded dataset. All malicious labels have been grouped together meaning that this becomes a binary classification problem. If the target variable produces a positive result then this means there has been some form of intrusion, however we cannot determine what type of attack has been performed.

Table 1 Original Class Distribution in the Downloaded RT-IoT2022 File.

Original Label	Binary Target	Records
DOS_SYN_Hping	Attack	94,659
Thing_Speak	Benign	8,108
ARP_poisioning	Attack	7,750
MQTT_Publish	Benign	4,146
NMAP_UDP_SCAN	Attack	2,590
NMAP_XMAS_TREE_SCAN	Attack	2,010
NMAP_OS_DETECTION	Attack	2,000
NMAP_TCP_scan	Attack	1,002
DDOS_Slowloris	Attack	534
Wipro_bulb	Benign	253
Metasploit_Brute_Force_SSH	Attack	37
NMAP_FIN_SCAN	Attack	28

➢ *Data Preprocessing*

When exporting the data, we removed the row index along with the id.orig_p and id.resp_p columns. The row index column was included for ordering purposes. The port numbers can affect the result significantly depending on how the service is set up. The resultant predictor table includes 79 numerical variables, and two categorical variables proto and service. Rows with identical predictor signatures were identified through a hash of the row. Any rows with conflicting binary labels were removed (503). Of the remaining rows there were 104,349 duplicates with identical signatures. We only kept one instance of each signature giving us a total of 18,265 unique and clear predictor records. Of these 11,952 are benign and 6,313 are attacks.

Removing the port numbers means that when using this representation of predictors for the audit we may have multiple instances being reduced to one based on their signature. As such our sample is a benchmark for the signatures rather than how often they would occur on a real gateway. By doing this prior to splitting we ensure that there isn't an exact match for the chosen feature vector between our training and test sets. We split the data using stratified sampling with a random seed value of 42 giving us a training set of 12,785, validation set of 2,740 and test set of 2,740. No oversampling or synthetic attacks were created.

For each numeric value x, the transform $g(x) = sign(x)\, ln(1 + |x|)$ reduces the influence of highly skewed flow counts and rates. Standardization then uses $x' = [g(x) - \mu train] / \sigma train$, where μtrain and σtrain are estimated from training records only. Encoded values are clipped to the interval [−8, 8]. The categorical encoder is also fitted on the training partition, with unseen categories mapped to an all-zero indicator vector. This produces 92 encoded features per record. Any non-finite numeric value is replaced with zero before standardization; this is a preprocessing rule rather than an inferred label.

Table 2 Input Representation and Experimental Configuration.

Component	Configuration
Dataset	RT-IoT2022; 123,117 downloaded flow records
Audit unit	Exact selected predictor signature; contradictory signatures excluded
Cleaned records	18,265 unique predictor records
Partition sizes	12,785 training; 2,740 validation; 2,740 test
Input tensor	Batch × 1 flow × 92 encoded features
GRU configuration	32 hidden units; zero initial hidden state per flow
Output	One sigmoid probability; attack if probability ≥ 0.5
Optimization	Adam; learning rate 0.001; batch size 256
Training control	Maximum 45 epochs; validation patience 7; best checkpoint restored
Repetitions	Initialization seeds 42, 52 and 62; fixed partition
Execution	CPU; Python, NumPy, pandas and scikit-learn

➢ *The Proposed GRU Algorithm*

Let x_t denote an input vector and h_{t-1} the preceding hidden state. A GRU computes an update gate z_t, a reset gate r_t and a candidate state n_t. In the equations below, σ is the logistic sigmoid, tanh is the hyperbolic tangent and ⊙ is elementwise multiplication. Input matrices W map the 92 features into 32 hidden units; recurrent matrices U map

the previous 32-dimensional state into the same space; b denotes a bias vector.

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \quad (1)$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \quad (2)$$

$$n_t = \tanh(W_n x_t + U_n(r_t \odot h_{t-1}) + b_n) \quad (3)$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot n_t \quad (4)$$

$$\hat{p} = \sigma(w_o^T h_t + b_o); \quad \hat{y} = 1[\hat{p} \geq 0.5] \quad (5)$$

For this evaluation, each input has length one and $h_0 = 0$. Equations (1)–(4) therefore simplify exactly to $z = \sigma(W_z x + b_z), n = \tanh(W_n x + b_n)$ and $h = (1 - z) \odot n$. The reset branch and recurrent matrices have no effect for a zero initial state and are omitted from the numerical implementation. The effective classifier contains 5,985 trainable parameters: $2 \times (92 \times 32 + 32)$ for the update and candidate projections, plus 32 output weights and one output bias. This is the single-step specialization of the GRU equations, with learned gating across flow attributes. It does not obtain a temporal advantage from recurrent memory.

The sigmoid head maps the gated state to a score between zero and one. A positive score is converted into an intrusion alert at the selected threshold. The classifier can use combinations of packet rates, payload summaries, flag counts, activity intervals and service indicators. Its score is treated as a decision score; probability calibration is not established merely by using a sigmoid output.

➢ *Model Training*

The model minimizes class-weighted binary cross-entropy, $L = -N^{-1} \Sigma_i [w_1 y_i \ln \hat{p}_i + w_0(1 - y_i) \ln(1 - \hat{p}_i)]$, where N is the number of training records and $w = N/(2N)$. The class count N is computed from training labels. These weights give the two binary classes equal aggregate influence during optimization. Adam uses a learning rate of 0.001, moment coefficients 0.9 and 0.999, and numerical constant 10^{-8}. Each gradient component is clipped to [−1, 1]. Mini-batches contain 256 records, and the training order is reshuffled each epoch.

I trained the model for up to 45 epochs, with an early stopping patience of 7 epochs. The best checkpoints were determined based on the lowest validation cross-entropy without weighting. Three different seeds (42, 52, 62) were used to alter the initial weights and order of mini-batches. Seed 42 was selected as reference based on convention. It had the lowest validation loss at the final training epoch (45). The other checkpoints were at epochs 44 and 43.

The logistic regression model was trained with balanced class weight and a maximum of 400 iterations. The linear support vector machine was trained with balanced class weight and a maximum of 3000 iterations. The random forest model consisted of 150 trees, with a minimum of 2 samples per leaf, balanced class weights and seed 42. All models received the same 92 features. The decision function of the SVM model was transformed into a monotonic function using a sigmoid function to allow ranking. I did not calibrate the probabilities but used a decision threshold of 0. Baseline models were trained with each seed once.

➢ *Performance Evaluation*

A true positive (TP) is an attack record correctly flagged; a true negative (TN) is a benign record correctly accepted. A false positive (FP) is a benign flow flagged as an attack, and a false negative (FN) is an attack accepted as benign. The metrics are accuracy = (TP + TN)/(TP + TN + FP + FN), precision = TP/(TP + FP), recall = TP/(TP + FN), F1 = 2TP/(2TP + FP + FN), and false positive rate = FP/(TN + FP). AUROC measures ranking across thresholds, while average precision summarizes the precision–recall ranking. All primary counts use threshold 0.5.

III. RESULTS AND DISCUSSION

The test set contains 1,793 benign and 947 attack flows. Table 3 shows how the GRU performed against the baseline models. The GRU model correctly classified 2,701 out of 2,740 flows in the test set, improving on the logistic regression and linear SVM models. However, the random forest model was the best performing with 2,724 correct classifications. This demonstrates how having a smaller sized gated representation works well for binary screening tasks. However, the tree based ensemble model better captured the relationship between the features in the table.

Table 3 Held-Out Binary Intrusion Detection Results.

Model	Accuracy %	Precision	Recall	F1	AUROC
Logistic regression	97.66	0.9642	0.9683	0.9663	0.9957
Linear SVM	98.14	0.9686	0.9778	0.9732	0.9967
Random forest	99.42	0.9885	0.9947	0.9916	0.9997
GRU	98.58	0.9759	0.9831	0.9795	0.9994

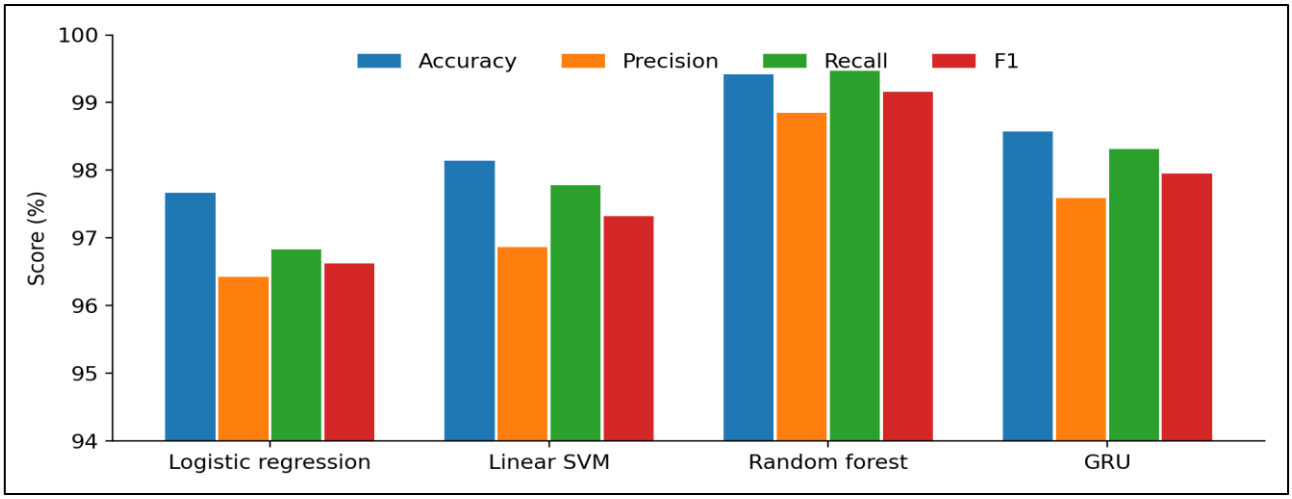


Fig 2 Measured Test-Set Accuracy, Precision, Recall and F1-Score of the Four Detectors.

GRU obtained better results than logistic regression (0.91) and linear SVM (0.44). The recall rate of attacks was also higher (98.31%) compared to logistic regression (96.83%) and linear SVM (97.78%). The model performed worse compared to random forest with an accuracy difference of -0.84 and attack recall difference of -1.16. However, this does not imply that it will always perform better than the other models in any other IoT environment, feature set or independent device collection.

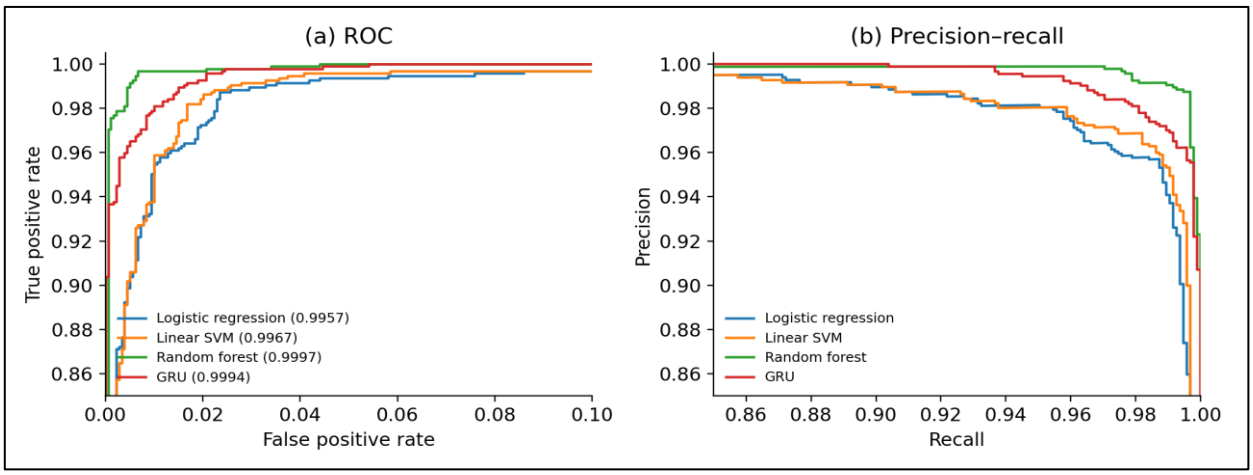


Fig 3 Measured ROC and Precision–Recall Curves; Panel (a) Enlarges the Low False-Positive Region.

The GRU achieved a high AUROC of 0.9994 and an average precision of 0.9988. This means that most attacks were ranked higher than benign traffic, however there are still 39 classification errors that occurred at the default operating point. Even with a near perfect ranking statistic, it is still important to look at what operating point you have selected. It can be seen from Figure 3 that all models rank the classes well. The random forest still has the best overall discrimination.

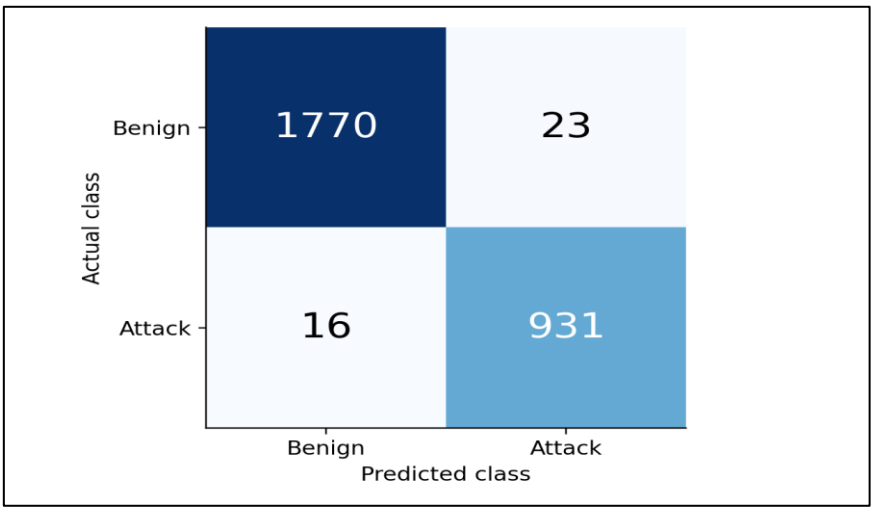


Fig 4 GRU Test Confusion Matrix at Threshold 0.5 for Seed 42.

The real results are shown in Figure 4. The system correctly identified 1770 legitimate records and 931 attack records. However, there were also 23 legitimate records that caused false positives and 16 attack records that were missed. This gave a false positive rate of 1.28% and false negative rate of 1.69%. Of the 954 positive alerts, 931 were correct. This gives the precision a value of 97.59%. Overall this shows how the summary values relate to the real numbers. If the proportion of attacks in operation changes then the precision may change even if the sensitivity and specificity stays the same.

Table 4 Per-Class GRU Results at the Reference Operating Point.

Class	Precision	Recall	F1	Support
Benign	0.9910	0.9872	0.9891	1793
Attack	0.9759	0.9831	0.9795	947

➢ *Decision Threshold Analysis*

Figure 5 and Table 5 show the operating characteristics of the fitted reference model at various threshold values. This is a hold out score operating characteristics. They do not represent an estimate of the optimal threshold (which is pre-specified as 0.5) but simply show how the model performs at different threshold values. Overall lower thresholds will increase the number of attacks and benign instances which are flagged. Threshold values should be selected from validation data based on your tolerance for alert volume. The model can then be evaluated on an independent operational sample.

Table 5 Descriptive Threshold Sweep for the Fitted Reference GRU.

Threshold	Recall %	FPR %	F1	FP	FN
0.3	99.47	2.06	0.9782	37	5
0.4	98.94	1.62	0.9796	29	10
0.5	98.31	1.28	0.9795	23	16
0.6	97.47	0.84	0.9793	15	24
0.7	96.73	0.67	0.9771	12	31

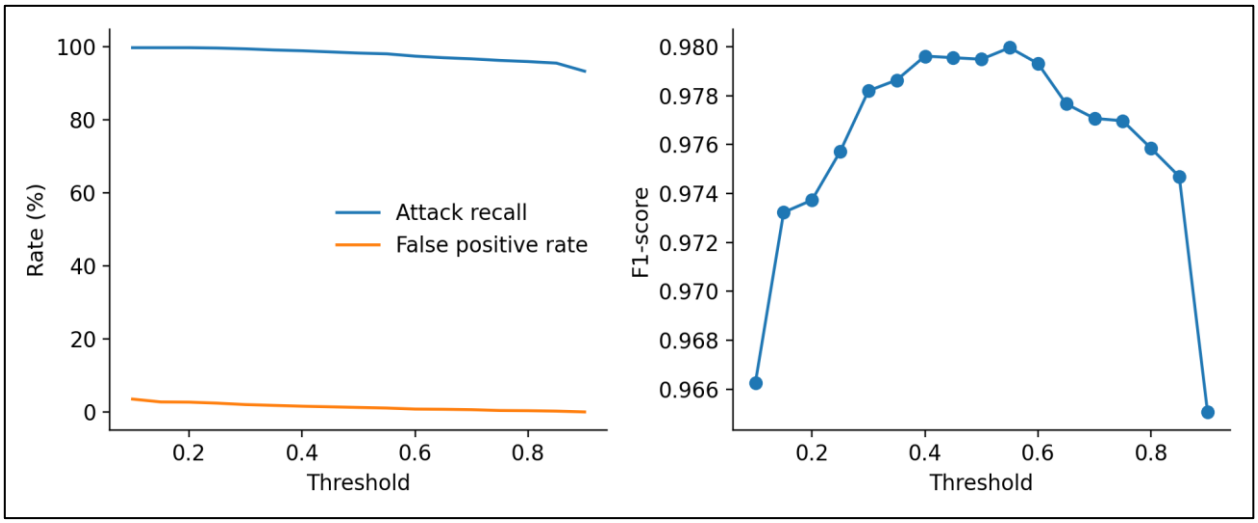


Fig 5 Attack Recall, False Positive Rate and F1-Score Across GRU Decision Thresholds.

➢ *Repeatability and Engineering Interpretation*

Table 6 GRU Repeatability Under Three Initialization Seeds.

Seed	Selected epoch	Accuracy %	F1	AUROC
42	45	98.58	0.9795	0.9994
52	44	98.58	0.9795	0.9994
62	43	98.65	0.9806	0.9994

The average accuracy across all three runs was 98.60%, with a sample standard deviation of 0.042%. The average F-Score over all three runs was 0.9799. The low standard deviation demonstrates there is little variance in the initialisation for this training test split. However, this does not indicate the uncertainty within different device populations. The reference run took around 4.72 seconds to complete the training loop that was implemented. This does not take into account download, cleaning and pre-processing time and is not an indication of the inference time taken on embedded devices.

The signature audit has had a large impact on the data used for this experiment. Only 18,265 of the original 123,117 rows in the dataset have been retained. Whilst this ensures that there are no repeated instances of chosen predictor variables between train and test sets, it also means that the accuracy will be weighted based on unique signatures present. Within this experiment, contradictory signatures have also been removed. These choices are part of how we have defined the problem and should be considered when trying to reproduce the experiment.

The input to the GRU layer does not contain source and destination addresses, the exported index and port numbers. This lessens the reliance on easily identifiable configuration variables. However, there is still information present within the protocol and service variables relating to how the data was collected. By resetting the hidden layer for each flow, there is no influence from the assumed ordering of files. If the model were deployed using confirmed ordered packets, we could test the multi-step GRU states. However, this would need to be a separate experiment to determine the accuracy and response delay.

This benchmark only tests known attack families using one public testbed and a stratified split. There is no testing for unknown device transfers, concept drift, encrypted payload understanding or zero-day attack identification. As there are few attack labels for the source, these have been grouped together within the attack class. Therefore, we cannot infer the recall for each individual attack type from the recall for the binary attack class. In future, the next stage of engineering validation should use a device or capture split, more IoT datasets and inference costs on gateways. Edge-IIoTset and CICIoT2023 are suitable additions to the benchmark (Ferrag et al., 2022; Neto et al., 2023).

IV. CONCLUSION

This study implemented a 32 unit GRU gate for binary intrusion detection of observed RT-IoT2022 flow features. A clear testing set was formed by performing signature auditing, one-time pre-processing and a static splitting of the dataset to form 18,265 records within the dataset. An initial GRU model was used to achieve an accuracy of 98.58%, an F1-score of 0.9795 and AUROC of 0.9994. This used three seeds for initialization and had an average accuracy of 98.60%. The best performing model was random forest which achieved an accuracy of 99.42%. The results demonstrate the GRU as a lightweight baseline model for flow classification tasks. It also shows the classical ensemble still performs very well. Future work should consider testing with known time series data, separate devices and considering the resource requirements of deployment.

REFERENCES

[1]. Adefemi, K. O., Mutanga, M. B., & Alimi, O. A. (2025). A hybrid CNN–GRU deep learning model for IoT network intrusion detection. Journal of Sensor and Actuator Networks, 14(5), 96. https://doi.org/10.3390/jsan14050096

[2]. Adnan, A., Muhammed, A., Abd Ghani, A. A., Abdullah, A., & Hakim, F. (2021). An intrusion detection system for the internet of things based on machine learning: review and challenges. Symmetry, 13(6), 1011. https://doi.org/10.3390/sym13061011

[3]. Airlangga, G. (2024). Comparative analysis of machine learning models for intrusion detection in internet of things networks using the RT-IoT2022 dataset. MALCOM: Indonesian Journal of Machine Learning and Computer Science, 4(2), 656–662. https://doi.org/10.57152/malcom.v4i2.1304

[4]. Banaamah, A. M., & Ahmad, I. (2022). Intrusion detection in IoT using deep learning. Sensors, 22(21), 8417. https://doi.org/10.3390/s22218417

[5]. El-Shafeiy, E., Elsayed, W. M., Elwahsh, H., Alsabaan, M., Ibrahem, M. I., & Elhady, G. F. (2024). Deep complex gated recurrent networks-based IoT network intrusion detection systems. Sensors, 24(18), 5933. https://doi.org/10.3390/s24185933

[6]. Ferrag, M. A., Friha, O., Hamouda, D., Maglaras, L., & Janicke, H. (2022). Edge-IIoTset: a new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning. IEEE Access, 10, 40281–40306. https://doi.org/10.1109/ACCESS.2022.3165809

[7]. Gyamfi, E., & Jurcut, A. (2022). Intrusion detection in internet of things systems: a review on design approaches leveraging multi-access edge computing, machine learning, and datasets. Sensors, 22(10), 3744. https://doi.org/10.3390/s22103744

[8]. Khan, M. A., Khan, M. A., Jan, S. U., Ahmad, J., Jamal, S. S., Shah, A. A., Pitropakis, N., & Buchanan, W. J. (2021). A deep learning-based intrusion detection system for MQTT enabled IoT. Sensors, 21(21), 7016. https://doi.org/10.3390/s21217016

[9]. Kilichev, D., Turimov, D., & Kim, W. (2024). Next–generation intrusion detection for IoT EVCS: integrating CNN, LSTM, and GRU models. Mathematics, 12(4), 571. https://doi.org/10.3390/math12040571

[10]. Neto, E. C. P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., & Ghorbani, A. A. (2023). CICIoT2023: a real-time dataset and benchmark for large-scale attacks in IoT environment. Sensors, 23(13), 5941. https://doi.org/10.3390/s23135941

[11]. Qaddoura, R., Al-Zoubi, A. M., Faris, H., & Almomani, I. (2021). A multi-layer classification approach for intrusion detection in IoT networks based on deep learning. Sensors, 21(9), 2987. https://doi.org/10.3390/s21092987

[12]. Sharmila, B. S., & Nagapadma, R. (2023a). Quantized autoencoder (QAE) intrusion detection system for anomaly detection in resource-constrained IoT devices using RT-IoT2022 dataset. Cybersecurity, 6(1), 41. https://doi.org/10.1186/s42400-023-00178-5

[13]. Sharmila, B. S., & Nagapadma, R. (2023b). RT-IoT2022 [Data set]. UCI Machine Learning Repository. https://doi.org/10.24432/C5P338
[14]. Waqas, A., Khan, S. D., Ullah, Z., Ullah, M., & Ullah, H. (2025). Comparative analysis of deep learning models for intrusion detection in IoT networks. Computers, 14(7), 283. https://doi.org/10.3390/computers14070283
[15]. Zhong, M., Zhou, Y., & Chen, G. (2021). Sequential model based intrusion detection system for IoT servers using deep learning methods. Sensors, 21(4), 1113. https://doi.org/10.3390/s21041113