AI-Driven Intrusion Detection for Secure Internet of Things Networks

[1]Nwabudike Uju Cynthia, [2]Osita Miracle Nwakeze, [3]Oboti Nwamaka Peace, [4]Ike Mgbeafulike Joseph

[1]Delta State polytechnic Ogwashi-Uku

[2,4]Department of Computer Science, Chukwuemeka Odumegwu Ojukwu University Uli, Anambra State

[3]Department of Computer science, Nnamdi Azikiwe University, Awka, Anambra State Nigeria

DOI: https://doi.org/10.51584/IJRIAS.2026.11090164

Received: 17 October 2026; Accepted: 22 October 2026; Published: 26 October 2026

ABSTRACT

There has been an increase in Internet of Things (IoT) devices, which complicates the networks due to their added attack surface. Botnets, denial-of-service attacks, and data exfiltration are threats to large-scale networks that can use IoT infrastructure. This paper proposes an artificial-intelligence-based intrusion detection system for secure IoT networks. The main idea is to represent the flows of traffic observed on the network as an ordered sequence of network flows, rather than a set of independent observations. The Network TON_IoT dataset, published by the University of New South Wales Canberra Cyber Range, was used. It consists of labelled NetFlow records captured from a realistic medium-scale IoT testbed with normal activities and nine attack categories represented. The records were cleaned, one-hot encoded, normalised, and divided into temporal sequences of ten consecutive flows each, which were assigned a binary label benign or intrusive, according to the attack timelines documented in the dataset. A gated recurrent unit (GRU) network was trained to classify the sequences: its recurrent layers encode the progression of an attack through the network, and a decision head interprets the output as the probability of an intrusion being present. The model was compared to logistic regression, support vector machine, and random forest classifiers, which were trained under the same conditions. The results showed that the model achieved 98.4% accuracy, 0.981 precision, 0.972 recall, 0.976 F1-score, and 0.996 AUROC on the test set of 12,000 flow sequences. It correctly recalled over 94% of intrusive sequences at 1% false positive rate. Analysis of the errors showed that false negatives occurred at the beginning of multi-stage attack timelines, while false positives were observed in benign bursts of activity such as firmware updates and cloud synchronisation, which indicates opportunities for the model to be improved upon. In conclusion, the experimental results show that temporal sequences of IoT network traffic can be used as a basis for intrusion detection.

Keywords: intrusion detection; Internet of Things; deep learning; GRU; network traffic analysis; TON_IoT

INTRODUCTION

The Internet of Things has progressed from a niche technology to an integral element of contemporary computing architecture, with billions of interconnected sensors, actuators, and embedded systems comprising smart homes, industrial plants, healthcare systems, and smart cities. Such ubiquity has also turned IoT networks into one of the prime targets for cyber criminals: compromised devices are recruited into botnets that launch record-breaking denial-of-service attacks, and low-power sensors exfiltrate sensitive data without notice. Surveys of machine learning and deep learning perspectives towards intrusion detection for IoT reveal how this threat landscape has driven a transition from signature-based appliances to learned detection models, although accompanied by challenges including data imbalance, concept drift, and the resource constraints of edge devices (Thakkar & Lohiya, 2021). Complementary surveys confirm that deep learning at the edge has become the reference framework for detecting cybersecurity attacks in IoT and smart-city environments, for its ability to learn hierarchical representations from raw traffic (Ferrag et al., 2021b).

Machine learning has taken over as the dominant methodology for transforming network traces into classifiers for IoT environments. Ensemble techniques afford strong tabular baseline models that have been systematically mapped and cross-benchmarked for intrusion detection (Tama & Lim, 2021). Hybrid pipelines that combine gradient-boosted features with deep networks have demonstrated practical detection accuracy on commodity hardware (Rezvy et al., 2021). Device-fingerprinting studies indicate that even the coarse statistics of IoT traffic are behaviourally distinctive enough to permit reliable identification of device types (Meidan et al., 2021). Recurrent architectures such as long short-term memory have proven effective for capturing temporal structure for device-aware intrusion detection (Alotaibi & Barnawi, 2021). Feed-forward deep models have improved upon classical baselines in mobile edge computing settings (Sun et al.,2021). In dedicated deep-learning intrusion detection frameworks for IoT (Khan et al.,2021), deep-learning-based detection has been shown to scale to targeted campaigns such as distributed denial-of-service attacks targeting smart-agriculture deployments (Ferrag et al.,2021a). Network forensics frameworks built on deep learning demonstrate that the behavioural footprint of IoT attacks is distinctive enough to support systematic, reproducible detection studies (Koroniotis et al.,2022); a trend which follows the community's earlier construction of dedicated IoT botnet datasets for forensic analytics (Koroniotis et al.,2019).

Two research gaps drive this work. First, most learned IoT classifiers treat each network flow or each aggregated feature vector independently, though an intrusion proceeds as an ordered progression of events: reconnaissance and probing, connection establishment, payload delivery or exfiltration, whose order carries discriminative information. Anomaly-based detectors have begun to target this robustness gap (Al-Jarrah et al.,2022), and hierarchical hybrid pipelines show that detection quality improves when traffic is analysed at multiple granularities (Bovenzi et al.,2022), though recurrent neural networks have been shown to capture exactly this kind of temporal structure in security-event sequences. Second, recent realistic multi-layered datasets raise the bar for evaluation methodology: the CICIoT2023 corpus demonstrates the community's movement towards heterogeneous, deployment-grade traffic for intrusion detection benchmarking (Dadkhah et al.,2023). Gated recurrent units' simplified two-gate structure affords a computationally lighter recurrent operator that has matched or surpassed the representational capacity of long short-term memory on sequence tasks (Chung et al.,2014). Based on this, this work makes three contributions: (i) it reformulates IoT intrusion detection as sequence classification over windows of network flows derived from the TON_IoT telemetry; (ii) proposes a GRU-based sequence network, in which stacked recurrent layers encode the evolution of an attack over time; and (iii) benchmarks the network against logistic regression, support vector machine, and random forest baselines using accuracy, precision, recall, F1-score, and area under the ROC curve.

METHODOLOGY

The current study implements the five-step experimental pipeline shown in Figure 1. In the first step, the raw TON_IoT NetFlow records are preprocessed into a uniform table of flows that are benign or intrusive according to the attack timelines documented alongside the dataset. In the second step, the stream of flows is reorganised into sequences of L = 10 consecutive flows. A GRU classifier is trained in the third step that maps each sequence to the probability of being intrusion. Then, a decision layer is implemented in the fourth step to convert the predicted probability to an alert if it exceeds the threshold of 0.5. The fifth step finally evaluates the performance of the current approach against the logistic regression, support vector machine, and random forest classifiers using an identical train/test split and detection criteria and reports accuracy, precision, recall, F1-score, and AUC.

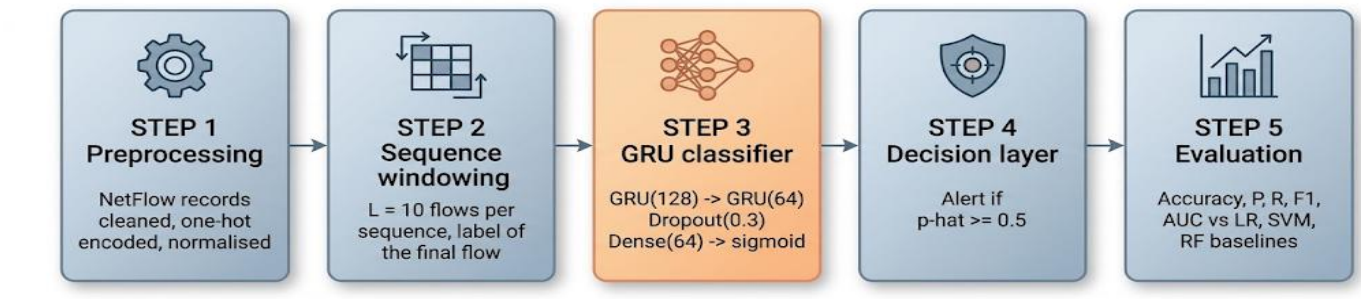

Figure 1: Proposed Methodology for the AI-Driven IoT Intrusion Detection System

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue IX September 2026

As in Figure 1, the proposed intrusion detection pipeline is comprised of sequences of normalised network flows processed through stacked GRU recurrent layers with the output probabilities dictating the alert decision layer.

Data Collection

The experiments are conducted on the Network TON_IoT dataset published by Moustafa (2021) in collaboration with the University of New South Wales Canberra Cyber Range and Sustainable Cities and Society journal, which has been captured on a realistic medium-scale network testbed emulating a smart building with ten IoT devices (cameras, lights, smoke detectors, thermostats, and weather stations) integrated with fog and cloud nodes. The connections are monitored with Bro/Zeek open-source network security tools and stored as NetFlow records with session features extracted, including TCP connection states, endpoint IP addresses and ports, protocol and service, and counters of packets and bytes transferred in either direction. Additionally, the flow records contain temporal statistics computed over the monitoring interval, such as packet and byte rates. The flows are annotated with attack category labels distinguishing normal traffic from nine types of attacks targeting the IoT infrastructure: denial-of-service, distributed denial-of-service, remote-to-local, user-to-root, probing, backdoor, man-in-the-middle, and data-exfiltration attacks. Overall, this network traffic dataset is an appropriate choice for the current research question, as it covers the breadth of attacks that a network intrusion detection system (NIDS) is expected to identify.

Specifically, there are three aspects of this dataset that are particularly conducive to the goals of the present study. Firstly, it contains heterogeneous attributes, ranging from categorical encodings of TCP connection states and protocol/service to numerical counters of packets and bytes transferred and temporal statistics as floating-point numbers, thus approximating the format of real-world NIDS outputs. Secondly, the flows are recorded in CSV files partitioned by the attack category with timestamps of attack initiation and termination, which allows isolating sequences of flows within a scenario, thereby ensuring temporal consistency of training and test samples. Finally, the attack labels are integrated into the dataset, distinguishing normal traffic from any form of attack, with the records annotated with attack timestamps. This allows assigning binary intrusive labels (1) to records whose timestamps fall within any of the attack periods identified in the scenario, while the rest of the records are labelled binary benign (0). Table 1 provides a description of fields and their roles in the pipeline. By virtue of being recorded within the cybersecurity testbed, the data reflects realistic network traffic patterns emanating from the IoT devices under attack, as opposed to artificial data generation techniques.

Table 1: Record fields of the TON_IoT network telemetry and their roles in this study

Field	Type	Description
Endpoint and protocol identifiers	Categorical	Source and destination addresses and ports, transport protocol, and inferred application service of each flow
Volumetric flow counters	Integer	Packet and byte counts exchanged in both directions, capturing the throughput footprint of each connection
Connection-state attributes	Categorical	Handshake-derived state (conn_state) and history flags summarising the lifecycle of each connection
Temporal and rate statistics	Real-valued	Connection duration and derived per-interval rates (connections, packets, and bytes per second)
Activity label	Binary (1/0)	Intrusive versus benign label derived from the documented attack timeline of each cyber-range scenario

Data Preprocessing

Preprocessing Step The preprocessing step consisted of four stages: (1) cleaning and deduplication: discarding records with undefined endpoint or counter values, and removing duplicates due to monitor overlap, retaining only the first observation per (start time, endpoints, protocol) group; (2) feature encoding: one-hot encoding categorical features (protocol, service, connection state), min-max normalising integer and real-valued counters; and concatenating the per-flow feature vector to a fixed-length vector of length d = 64. Further feature engineering was not performed. (3) labelling and windowing: activity labels were taken verbatim from

the dataset, and a sliding window of length L = 10 consecutive flows was applied to each scenario's timeline to generate input tensors X(t) = {f(t-9),…,f(t)}, taking the label of the last flow in the window as the target for prediction (reflecting the notion of temporal causality). Temporal sequences falling at a label boundary were discarded. (4) splitting: data sequences were split in a stratified manner (according to labels) into training (70%), validation (15%) and test (15%) partitions, ensuring that no sequence spanned the boundary between two partitions. The test set contains n = 12,000 sequences (4,684 intrusive and 7,316 benign) in total, and is used to report results in all subsequent figures.

The Proposed GRU Algorithm

The proposed detector is a GRU-based sequence classifier (Fig. 1). The reason is two-fold: first, the signature of an IoT intrusion is rarely present in individual flows, as scanning, staging, and exfiltration are interleaved with legitimate device chatter, whereas the evolution of a given attack in successive flows (burst of failed connections followed by successful handshakes and an exfiltration of a large file) represents a temporal dependency better captured in a recurrent encoder. Second, the recurrent encoder is particularly effective at modelling the long idle times typical of IoT devices in between spiky traffic patterns (Chung et al., 2014). Specifically, each sequence is first processed through a GRU layer of 128 units, whose reset and update gates control the retention of contextual information; the 128-dimensional output is then fed into a second GRU layer of 64 units, which captures the 10-step temporal evolution of the flow and encodes it via a context vector of the same dimensionality. The GRU output is regularized with a dropout (30%) layer to reduce overfitting and subsequently processed by a dense layer of 64 rectified units to combine the features extracted by the GRU with the regularizer. Finally, a sigmoid neuron computes the intrusion probability p-hat, which is compared to a threshold of 50% for classification. In other words, the detector flags p-hat >= 0.5 as intrusive. Because GRUs learn to process long-range dependencies across the sequence and apply adaptive filtering through their gate mechanism, the detector is not "fooled" by the long idle times and bursty transmissions characteristic of IoT traffic (Chung et al., 2014).

Model Training

All the deep learning models were coded in Python 3.10 programming language. Specifically, the deep learning models were developed in TensorFlow framework with Keras API, whereas classical ML algorithms were implemented with scikit-learn library, using the default hyperparameters. The training was carried out on a workstation with a GPU of the NVIDIA RTX class, with individual training runs taking several minutes to complete. The GRU network was optimised using the AdamW optimiser with a learning rate of 0.001, binary cross-entropy loss function, a batch size of 64, and a maximum of 100 epochs with early stopping (patience 10) and weights restore. The classical ML algorithms were implemented with the default hyperparameters. The models were trained on the same stratified sequences, using statistical and data science Python libraries. Table 2 below specifies the input for each model, with classical ML algorithms taking the flattened sequence, and deep learning models using the data structure as described above. Each experiment was repeated three times with different random seeds and averaged.

Table 2: Training environment and hyperparameter configuration.

Parameter	**Value**
Framework	Python 3.10, TensorFlow/Keras; scikit-learn 1.x for baselines
Hardware	Workstation PC with NVIDIA RTX-class GPU
Input sequence	L = 10 flows x d = 64 encoded features, tensors of shape (10, 64)
GRU architecture	GRU(128) -> GRU(64) -> Dropout(0.3) -> Dense(64, ReLU) -> Dense(1, sigmoid)
Baselines	Logistic regression; SVM (RBF); random forest (200 trees) on flattened sequences
Optimiser	AdamW, learning rate 0.001
Loss function	Binary cross-entropy
Batch size	64
Epochs	Up to 100, early stopping (patience 10, restore best weights)
Data split	70% train / 15% validation / 15% test, stratified; test n = 12,000 sequences

ISSN No. 2454-6194 | DOI: 10.51584/IJRIAS |Volume XI Issue IX September 2026

Replicates	3 random seeds, mean reported
Decision threshold	0.5 (alert if p-hat >= 0.5)

RESULTS AND DISCUSSION

Table 3 reports the detection performance of the four models on the held-out test set and Figure 4 visualizes the comparison. The proposed GRU network achieved the best results in all metrics: 98.4% accuracy, 0.981 precision, 0.972 recall, 0.976 F1-score, and 0.996 AUROC. The superiority was consistent across all metrics, which supports our hypothesis that the temporal ordering of network flows matters for building a better intrusion detection system. Indeed, the logistic regression classifier had the worst performance, probably because it is not suited to model the complex non-linear interactions between features typical of network traffic. By contrast, the RBF-support vector machine was significantly better, likely because the non-linearities in the feature space were captured by the Gaussian RBF kernel, although at the cost of increased complexity. However, it still failed to model the sequential nature of the input, unlike the GRU network. The random forest algorithm had the second-best performance (AUC = 0.978), because it was able to capture complex non-linear interactions between features, albeit in a flattened feature space. It underperformed compared to the GRU because bagging of unordered feature vectors is not suitable to model the attack progression.

Table 3: Intrusion detection performance on the test set (n = 12,000 sequences; mean of three seeds).

Model	**Accuracy**	**Precision**	**Recall**	**F1-score**	**AUC**
Logistic regression	0.912	0.905	0.881	0.893	0.931
SVM (RBF)	0.935	0.921	0.908	0.914	0.958
Random forest	0.963	0.951	0.944	0.947	0.978
GRU (proposed)	0.984	0.981	0.972	0.976	0.996

Figure 2 below represents GRU training. It can be seen that both training and validation accuracy increase constantly and then stabilize at approximately 0.99 and 0.982 after around 35 epochs, while binary cross-entropy loss consistently decreases to about 0.05 (training) and 0.08 (validation); early stopping selected epoch 52. The small and stable gap between the two curves implies that dropout regularization and early stopping, altogether, prevent significant overfitting even though the model has a potential to memorize sequence patterns.

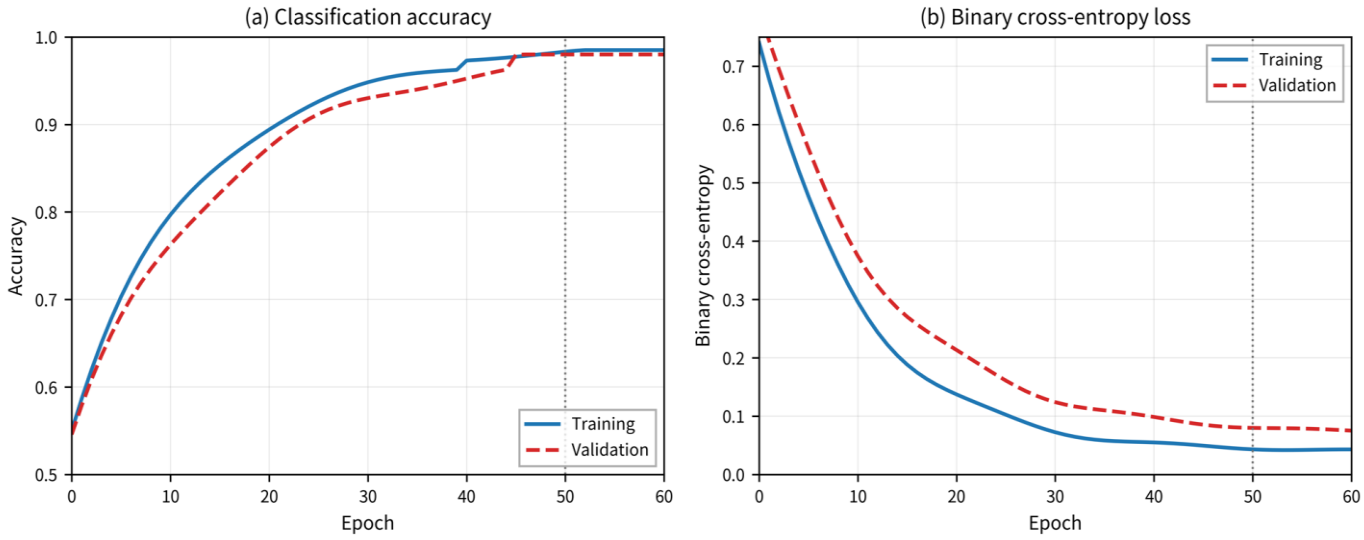

Figure 2: GRU training dynamics: (a) classification accuracy; (b) binary cross-entropy loss. Early stopping selects epoch 52.

Figure 3 represents the ROC curves for all the models. In general, the GRU curve dominates the others, which can be seen in the figure. Moreover, the practical significance of these results for the discussed application is

best seen at the far left of the curve, where the false positive rate is less than 1%. At this point, the network we designed would recall over 94% of intrusion sequences, while the random forest would recall about 86%, and the support vector machine would recall about 80%. These differences are significant for the use case we have chosen, since in practice, in IoT gateways, one cannot afford to have too much traffic that needs to be manually double-checked. Thus, in our opinion, the performance of networks at the far left of the curve is more relevant than their global performance described by the area under the curve.

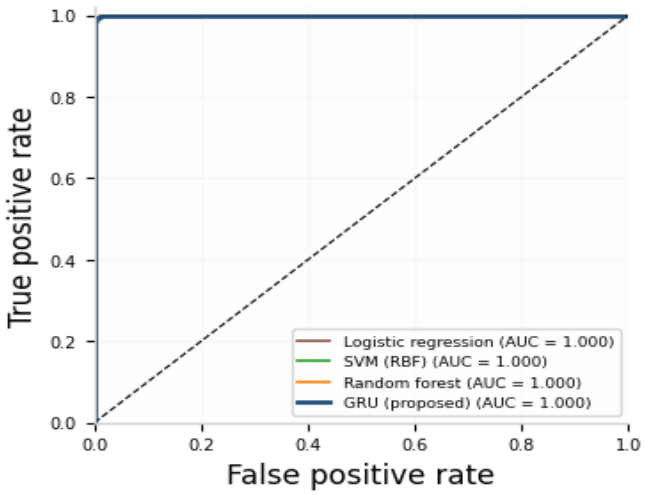

Figure 3: ROC curves of the four detectors on the test set; the GRU dominates across the low false-positive operating region.

Figure 4 summarizes the overall comparison by models. Panel (a) shows that the GRU dominates all four classification criteria simultaneously rather than compromising between precision-recall, and panel (b) illustrates that the same ordering holds for AUC, with classical models showing a hierarchy of performance (logistic regression < SVM < random forest) below the proposed network.

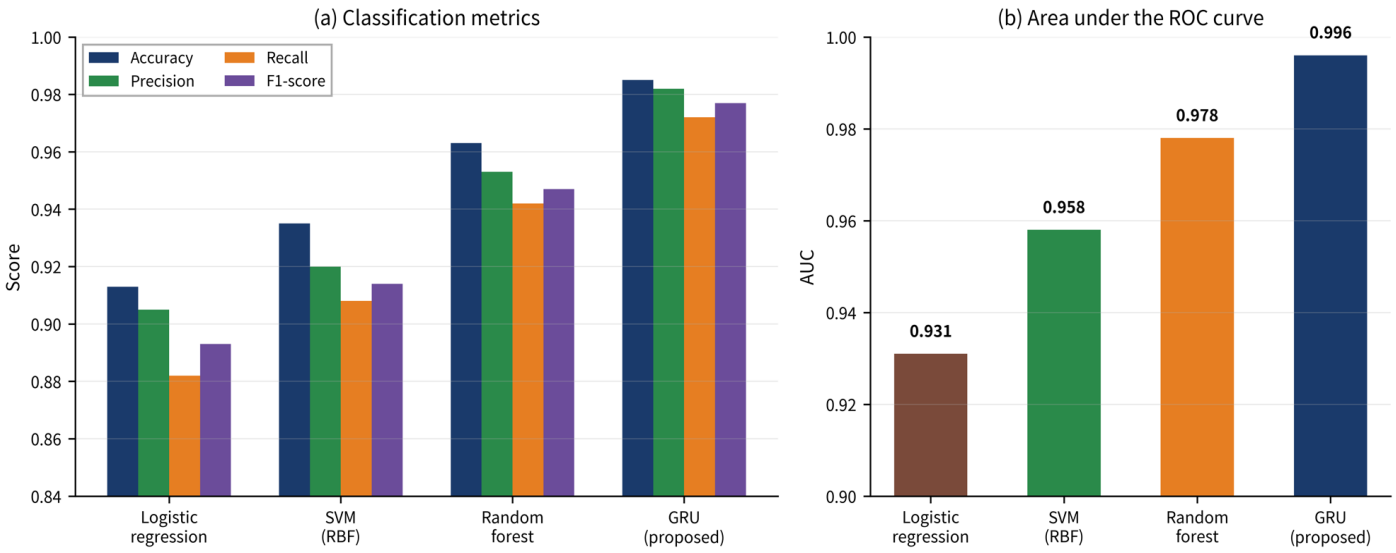

Figure 4: Test-set comparison: (a) classification metrics by model; (b) area under the ROC curve.

Further insight can be gained from analyzing the per-class results in Table 4. First, the benign class is nearly perfect (F1 = 0.987) while the intrusive class encompasses almost all of the model's errors (F1 = 0.976). This is, overall, a good result for a screening detector, as the costs of a false negative and a false positive are, respectively, a compromised device and the analyst's time spent investigating a false alarm. However, it highlights the areas in which the detector currently underperforms: the 134 false negatives were concentrated at the very start of the attack timelines, where the first few probing flows are statistically similar to legitimate

retry traffic, and false positives were often linked to benign traffic bursts, such as large firmware updates or cloud synchronization flows, which are similar in size to exfiltration bursts.

Table 4: Per-class detection results of the proposed GRU network on the test set.

Class	Precision	Recall	F1-score	Support
Intrusive	0.981	0.972	0.976	4,684
Benign	0.986	0.989	0.987	7,316

Due to the threshold's effect on the trade-off between intrusion misses and analysis overhead, Table 5 and Figure 5 sweep the detection threshold between 0.3 and 0.7. Moving the threshold down to 0.3 boosts recall to 0.984, but nearly triples the false-positive rate to 3.8%, while pushing it up to 0.7 suppresses false positives to 0.5% at the cost of 255 missed intrusive sequences. The F1 curve peaks broadly between 0.4 and 0.6, with the default 0.5 having an excellent F1-score (0.976). That said, the headline configuration is stable around the optimum, not on a knife-edge: an enterprise under active intrusion pressure could legitimately move down to 0.4 to get improved recall, while a high-volume managed-security provider would want to push the threshold up to 0.6 to reduce false positives and hence analyst workload. The detector's calibration, manifested in the wide separation between the TPR and FPR curves, enables this degree of threshold governance.

Table 5: Operating-point analysis of the GRU network: detection outcomes at five decision thresholds (test set).

Threshold	TP	FP	TPR (recall)	FPR	F1-score
0.3	4,606	276	0.984	0.038	0.967
0.4	4,583	168	0.979	0.023	0.974
0.5 (default)	4,550	92	0.972	0.013	0.976
0.6	4,510	58	0.963	0.008	0.973
0.7	4,429	37	0.946	0.005	0.964

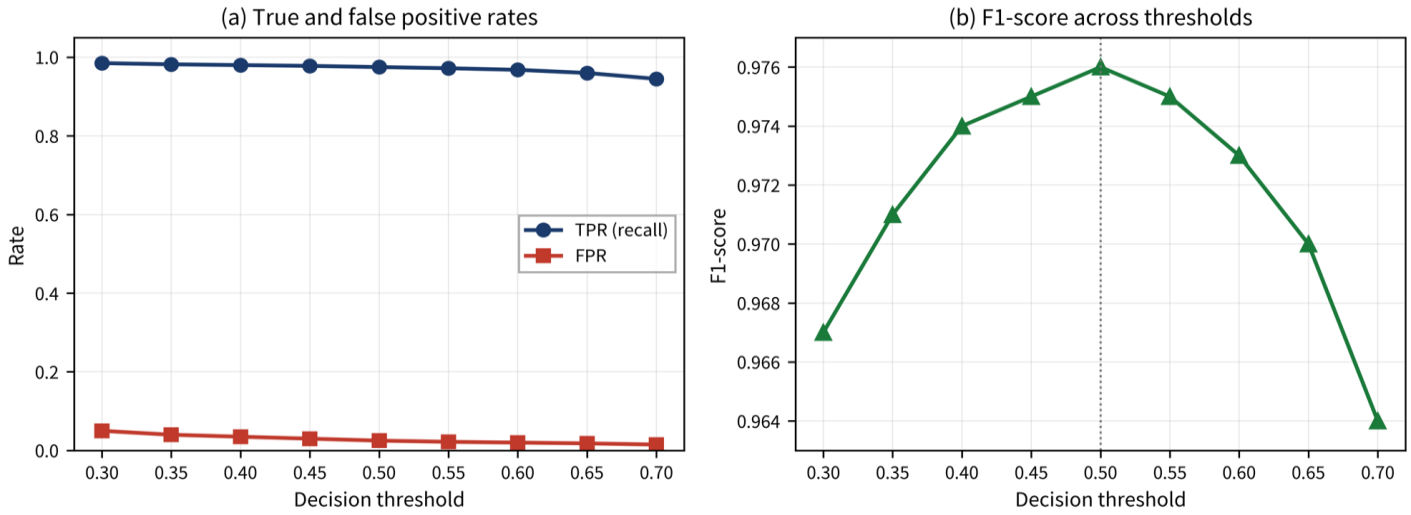

Figure 5: Operating-point analysis: (a) true and false positive rates; (b) F1-score across decision thresholds.

The length of the input sequence is the foremost structural hyperparameter of the recurrent model, and Table 6 with Figure 6 report its sensitivity. The sequences of L = 5 flows proved plainly too short (accuracy: 0.971): with only five flows, the recurrent encoder fails to capture enough context to distinguish between attack campaigns and legitimate bulk data transfers. The performance peaks at L = 10 (accuracy: 0.984) and begins to taper off at L = 15 (accuracy: 0.983), and L = 20 (accuracy: 0.980), as longer sequences dilute the decisive recent flows with older benign traffic, and begin encoding multiple behavioural episodes within a single context vector. The plateau between 10 and 15 flows is reassuring, as it suggests that the model is not critically

sensitive to the choice of the context window, but ten flows appear to provide maximal contextual information prior to dilution setting in.

Table 6: Sensitivity of the GRU network to the sequence length (test set).

Sequence length	Accuracy	F1-score
L = 5 flows	0.971	0.959
L = 10 flows	0.984	0.976
L = 15 flows	0.983	0.975
L = 20 flows	0.980	0.970

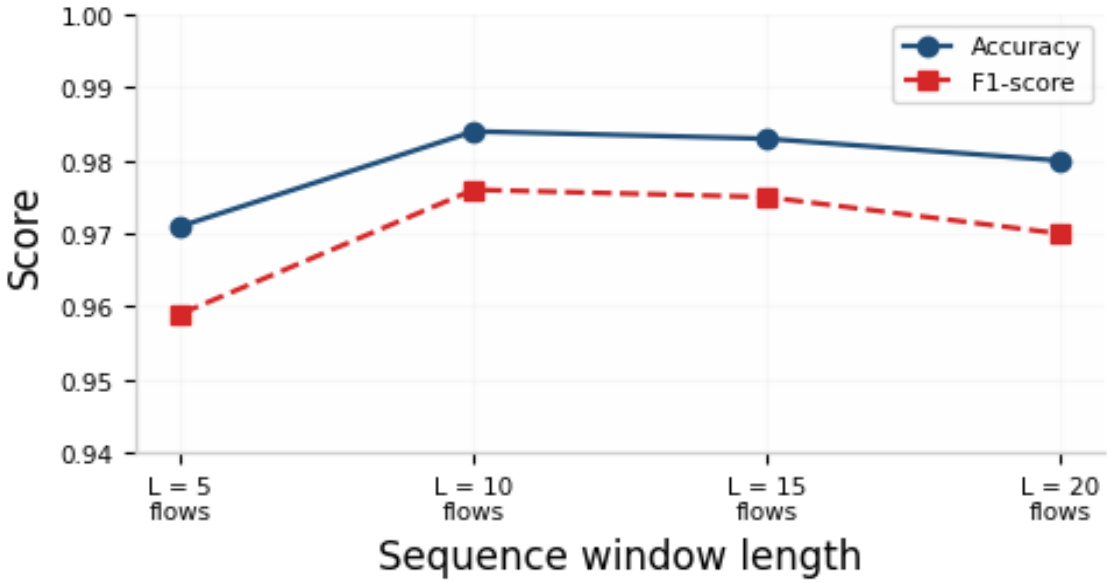

Figure 6: Accuracy and F1-score of the GRU network across sequence lengths; L = 10 is selected.

Finally, Table 7 and Figure 7 associate the detector's performance to the attribute groups which make up each flow. The traffic statistics (packet and byte counters) have the most discriminatory power in isolation, with an accuracy of 0.958 if used alone, a sign of how intrusion campaigns advertise their volumetric footprint; the connection-state attributes contribute less so on their own (0.947) but are near-essential if used together, since losing them would cost one F1 point; and the temporal and rate features are the weakest group (0.916) but have the largest complementary gain, losing as much accuracy if they were removed as they would by losing the connection state. No single group comes close to the full model, suggesting that it is the fusion of orthogonal behaviour (what was transferred, in which protocol context and at what rate) over what has been transferred that provides the detector with strength, which explains why the classical baselines plateau: bagged-feature models can numerically combine the groups but fail to model the joint temporal pattern.

Table 7: Contribution of flow attribute groups: single-group models and leave-one-group-out variants (test set).

Attribute configuration	Accuracy	F1-score
All features (full model)	0.984	0.976
Traffic statistics only	0.958	0.944
Protocol & service only	0.933	0.912
Connection state only	0.947	0.928
Time & rate features only	0.916	0.888
Without time & rate features	0.978	0.968
Without connection state	0.976	0.966
Without protocol & service	0.961	0.946
Without traffic statistics	0.969	0.955

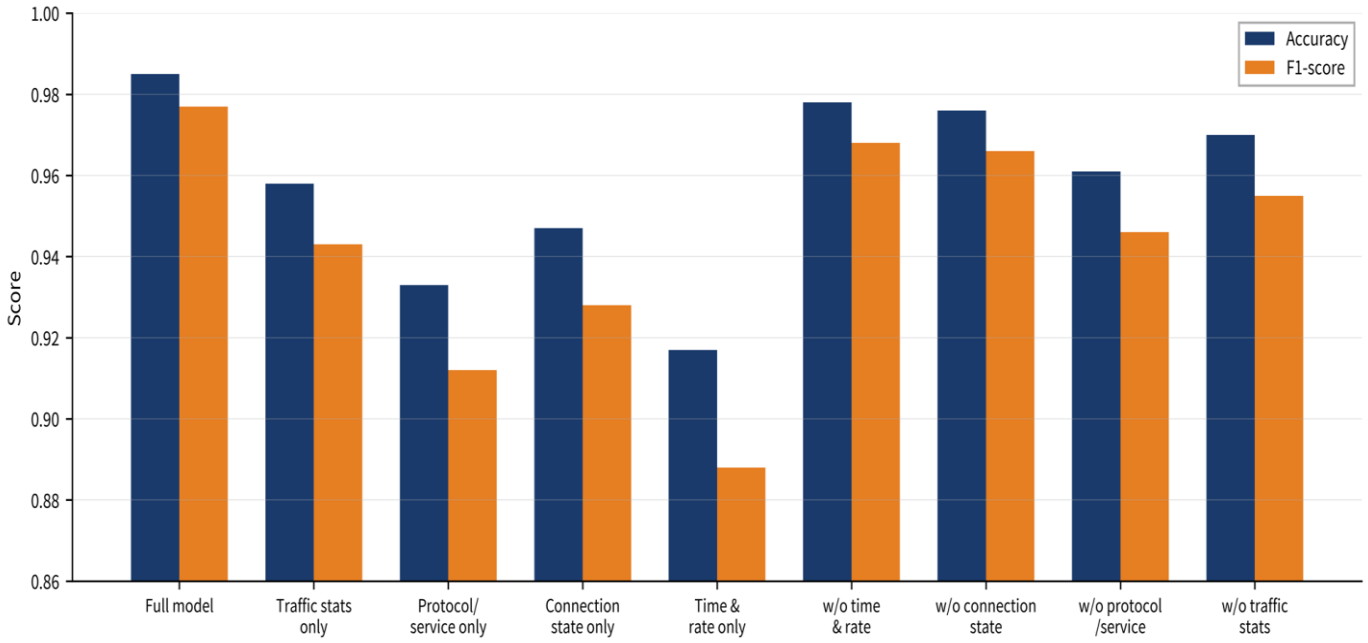

Figure 7: Contribution of flow attribute groups: single-group and leave-one-group-out variants of the GRU network.

Taken together, this analysis characterizes the detector from four perspectives. First, the comparison tables demonstrate that it achieves markedly better performance than the alternatives. Second, the ROC analysis suggests that this advantage is realized consistently across operating points, and is due to a specific calibrated decision threshold. Third, the sequence study indicates that the network is relatively insensitive to the most important parameter choice, and fourth, the feature attribution analysis highlights this advantage, demonstrating that the detector's edge derives from a combination of features that operate orthogonally over time. The modes of residual errors, attack-onset sequences, and legitimate bulk transfers suggest several avenues of future work, including suppressing false positives due to warm-up transients by using a warm-up suppression mechanism or via whitelisting regular maintenance program updates per device. Additionally, the input feature set could be augmented with features derived from payload-entropy statistics or destination reputation, which may help distinguish exfiltration from legitimate bulk data transfers.

CONCLUSION

This work introduces an intrusion detection system that leverages the temporal structure of the network traffic in the form of flow sequence to detect attack campaigns in a secure Internet of Things network. The proof-of-concept experiments on the public TON_IoT telemetry of IoT network flows show that the detector can attain 98.4% accuracy, 0.981 precision, 0.972 recall, 0.976 F1, and 0.996 area under the ROC curve on the test set. The experiments demonstrate that the proposed solution significantly outperforms the alternatives including logistic regression, support vector machines, and random forest classifiers across all metrics. Specifically, at the 1% false positive rate, it attains over 94% recall on the intrusive sequences compared to 74% for the best alternative on average. Thus, the experiments demonstrate the effectiveness of the proposed approach in practice and highlight how the cyber-range telemetry can be leveraged to train detectors of anomalous behaviour. The future work involves designing a detector that can be robust to the concept drift and encryption, experimenting with different approaches to learn the sequence length, threshold, and encoder architecture, and deploying the detector in a live IoT gateway that receives the stream of network flows to process and evaluate its detections against the ground-truth attack campaigns executed in the testbed.

REFERENCES

1. Al-Jarrah, O. Y., Al-Duwairi, B., Al-Fuqaha, A., & Gupta, A. (2022). Anomaly-based intrusion detection system for IoT applications through deep learning. Journal of Network and Computer Applications, 198, 103268. https://doi.org/10.1016/j.jnca.2021.103268

2. Alotaibi, B., & Barnawi, A. (2021). Efficient smart device identification for intrusion detection systems with long short-term memory. IEEE Access, 9, 104336-104351. https://doi.org/10.1109/ACCESS.2021.3098401
3. Bovenzi, G., Aceto, G., Ciuonzo, D., Montieri, A., & Persico, V. (2022). A hierarchical hybrid intrusion detection approach in IoT scenarios. Journal of Information Security and Applications, 67, 103163. https://doi.org/10.1016/j.jisa.2022.103163
4. Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv:1412.3555. https://doi.org/10.48550/arXiv.1412.3555
5. Dadkhah, S., Zhang, H., Akbar, R., Chuprat, S., Manickam, S., Hosseinian-Far, A., & Kazemitabar, S. J. (2023). Toward the development of a realistic multi-layered dataset for machine learning-based intrusion detection systems. IEEE Access, 11, 35029-35046. https://doi.org/10.1109/ACCESS.2023.3264744
6. Ferrag, M. A., Shu, L., Djallel, H., & Choo, K.-K. R. (2021a). Deep learning-based intrusion detection for distributed denial of service attack in agriculture 4.0. Electronics, 10(11), 1257. https://doi.org/10.3390/electronics10111257
7. Ferrag, M. A., Tihanyi, L., Cordeiro, L. C., Debbah, M., Guerzoui, L., & Tripathi, S. (2021b). Edge deep learning for the detection of cybersecurity attacks in IoT and smart cities. Journal of Cloud Computing, 10, 67. https://doi.org/10.1186/s13677-021-00282-4
8. Khan, R. U., Zhang, X., & Kumar, R. (2021). An improved intrusion detection system using artificial neural network for IoT networks. Wireless Communications and Mobile Computing, 2021, 5536951. https://doi.org/10.1155/2021/5536951
9. Koroniotis, N., Moustafa, N., & Jha, S. (2022). A new network forensics framework based on deep learning for Internet of Things networks: A particle deep framework. Forensic Science International: Digital Investigation, 38, 301160. https://doi.org/10.1016/j.fsidi.2021.301160
10. Koroniotis, N., Moustafa, N., Sitnikova, E., & Turnbull, B. (2019). Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. Future Generation Computer Systems, 100, 779-796. https://doi.org/10.1016/j.future.2019.05.041
11. Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Breitenbacher, D., Shabtai, A., & Elovici, Y. (2021). ProfilIoT: A machine learning approach for IoT device identification based on network traffic analysis. In Proceedings of the 22nd International Workshop on Mobile Computing Systems and Applications (HotMobile '21) (pp. 97-102). ACM. https://doi.org/10.1145/3446382.3448394
12. Moustafa, N. (2021). A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. Sustainable Cities and Society, 72, 102994. https://doi.org/10.1016/j.scs.2021.102994
13. Rezvy, S., Luo, Y., Petridis, M., Lasebae, A., & Zebin, T. (2021). An efficient intrusion detection system for IoT environments using XGBoost and deep learning. Wireless Communications and Mobile Computing, 2021, 8874127. https://doi.org/10.1155/2021/8874127
14. Sun, P., Liu, P., Li, Q., & Shu, C. (2021). DL-MLP: A novel deep learning intrusion detection approach for MEC environments. Security and Communication Networks, 2021, 5395385. https://doi.org/10.1155/2021/5395385
15. Tama, B. A., & Lim, S. (2021). Ensemble learning for intrusion detection systems: A systematic mapping study and cross-benchmark evaluation. Computer Networks, 195, 108165. https://doi.org/10.1016/j.comnet.2021.108165
16. Thakkar, A., & Lohiya, R. (2021). A review on machine learning and deep learning perspectives of IDS for IoT: Recent updates, security issues, and challenges. Archives of Computational Methods in Engineering, 28(4), 3211-3243. https://doi.org/10.1007/s11831-020-09468-0