An Evaluation of AI-Based Network Intrusion Detection in Resource-Constrained Environments

Syed Rizvi*, Mark Scanlon†, Jimmy McGibney*, John Sheppard*
*South East Technological University, Waterford, Ireland
Syed.Rizvi@postgrad.wit.ie, {Jimmy.McGibney, John.Sheppard}@setu.ie
†School of Computer Science, University College Dublin, Ireland
mark.scanlon@ucd.ie

***Abstract*—Internet of Things (IoT) and edge computing devices have become integral to corporate and industrial systems. These devices are prime targets for attackers due to their constant availability and potential access to sensitive data. Handling substantial data quantities, these devices pose challenges in identifying relevant forensic evidence and investigating abnormal activities. Thus, accurate network intrusion detection is crucial in these resource-constrained environments. In addition, robust IoT forensic readiness strategies are vital for effective investigation. Unlike traditional computer forensic readiness, these strategies adapt to heterogeneous architectures. This paper evaluates an approach that directly trains and deploys AI models on resource-constrained devices, securing networks and categorizing significant traffic for later investigation. The approach identifies and records potential malicious attacks in real-time with minimal overhead, suitable for constrained environments. The experimentation employed the IoT-23 dataset. The outcome of the evaluation revealed that each of the included algorithms achieved a classification accuracy of over 99% on a representative resource-constrained device.**

***Index Terms*—Network Intrusion Detection Systems, Resource Constrained Environments, Internet of Things, Artificial Intelligence**

I. Introduction

The integration of IoT devices and Edge computing has transformed communication among smart devices, spanning applications from autonomous vehicles to medical implants [1]. These resource-constrained environments (RCEs) establish direct or internet-based API-driven connections, creating a network of diverse edge and IoT devices. Robust cloud servers manage these devices, enhancing the capabilities of low-powered gadgets. However, the heterogeneity of these devices introduces challenges to conventional computer security.

Modern network investigations handle large data volumes [2]. With the growth of affordable, lightweight devices, this network traffic data influx is expected to continuously increase – further compounding the challenges to store and index this data [3]. Cyberattacks take various forms, targeting abnormal assets and impacting individual or multiple devices serving as platforms for these attacks [4]. Securing edge devices and establishing intrusion-resistant networks are crucial to maintaining data integrity, streamlining digital evidence collection, and reducing investigative costs.

The rise of IoT presents challenges in terms of bandwidth and service provision due to high device-generated traffic. Solely relying on cloud computing for IoT applications with time-sensitive requirements is problematic [5]. A shift to distributed edge computing is occurring, where edge paradigms replace centralized models in IoT architectures. These frameworks use distributed edge resources to provide timely and context-aware IoT services. Architectures combining fog and edge computing bridge the cloud-IoT gap [6]. Security is pivotal for these often vulnerable, lightweight devices.

Deep Learning (DL) effectively detects malicious activities and aids in network forensics [7]. However, deploying DL models on RCEs poses computational and storage challenges. This study introduces a resource-constrained network forensic approach, extending DL capabilities to RCE devices. This approach minimizes computational and storage requirements while ensuring accurate identification of network attacks. By adopting this approach, IoT/edge devices enhance network forensic analysis, bolstering overall edge system security.

This paper conducts a comprehensive study on nine AI-driven intrusion detection models for multi-classification tasks. The models are evaluated using the IoT-23 dataset [8]. This research introduces lightweight methodologies for Network Intrusion Detection Systems (NIDSs) in RCEs, effectively classifying various attack types. The proposed approach detects attacks with limited training data, using less complex model architectures. The resulting models demonstrate notable improvements, reducing false positives and false negatives, thus enhancing accuracy in identifying network attacks.

II. Related Work

DL's scalability and automated feature creation capabilities make it a valuable asset in NIDSs and forensic readiness [9]. By leveraging DL, accurate representations can be extracted from extensive network data, enabling effective analysis of traffic patterns. These advancements enhance security measures and address the unique requirements of systems with resource limitations.

The objective of IoT forensic readiness is to establish protocols for identifying, acquiring, and analyzing IoT data within an organization. This includes identifying critical data sources and developing a strategic plan for managing forensic data during incident response. Kebande et al. [10] propose a proactive framework for meeting IoT requirements, implementing ISO/IEC 2704 standards [11]. The framework addresses IoT

forensic readiness in planning and implementation, incorporating organizational readiness and IoT security processes. By considering complexities and policy developments, it fills the void with a comprehensive IoT forensic readiness framework. However, it lacks low-level features for environment-specific customization without altering processes.

In Intrusion Detection Systems (IDSs) for RCEs, various approaches have been proposed. One is the "Edge-of-Things" (EoT) framework by Almogren [12], using a deep belief network to detect intrusive EoT events. However, this model requires significant computational resources. Idrissi et al. [13] implemented a DL-based host IDS using the MQTTIOT-IDS2020 dataset [14]. The dataset includes four attack types: Aggressive scan, UDP scan, Sparta SSH brute-force, and MQTT brute-force. Their proposed model, a Convolutional Neural Network (CNN), was tested on Raspberry Pi, Arduino, and ESPWROOM32 using TensorFlow Lite [15]. No universally optimal model performed best for all RCEs.

Djigal et al. [16] conducted a survey on the application of ML and DL for resource allocation in edge computing. Meanwhile, Saha et al. [17] delved into the intricate domain of lightweight ML algorithms, exploring TinnyML and ultra-lightweight counterparts like SqueezeNet and EfficientDET, specifically tailored for microcontroller-class hardware.

Another notable contribution emerged from the work of Kolcun et al. [18], who tackled the challenge of traffic classification in IoT. In their study, the authors harnessed the power of feature vectors as inputs for CNN. Among the proposed models, a standout candidate is the 4-Layer CDM. This model's uniqueness lies in its input structure, a vector derived from the authors' proprietary dataset. Comprising 19 carefully chosen features, including source and destination ports, received byte count, mean and variance of packet sizes, and stream duration, the model demonstrates exceptional promise in addressing the intricacies of traffic classification.

III. Methodology

The research presented as part of this paper employed the Design Science Research (DSR) methodology [19], which involves creating practical solutions to address real-world problems while contributing to scientific knowledge. DSR focuses on developing artifacts that serve human purposes, solving domain-specific issues, and assessing their value or utility. Consequently, extensive experiments were conducted, focusing on ML and DL techniques within RCEs. The objective was to explore various model architectures and hyperparameters to identify the optimal configurations that achieve high performance while minimizing computational power requirements.

A. Dataset Description

1) IoT-23: The IoT-23 dataset [8] is a recent resource for analyzing IoT traffic. It includes data from hardware IoT devices like Philips HUE, Amazon Echo, and Somfy Smart Door Lock. The dataset consists of 20 instances capturing malware executions on IoT devices and three instances of benign traffic. It covers various attack scenarios, including botnets like *Mirai* and *Torii*.

In addition to the packet capture files, the dataset provides Zeek/Bro IDS-generated netflows with contextual details. It includes labels describing flow nature, threat level, or involvement in malicious behaviors.These labels were generated during malware analysis in the Stratosphere Laboratory [20]. The IoT-23 dataset is valuable for researchers in IoT security and anomaly detection. It offers real-world IoT traffic for studying and understanding IoT network behavior.

IV. Experiments

A. Simulation Environments

1) Resource Slack Environment: The models were trained and tested in a Slack environment on a Windows OS, featuring an Intel Core i7-1165G7 processor and 8GB RAM. Development employed Python 3.9.7, alongside `Keras`, `TensorFlow`, and `Scikit-learn` for modeling, while data preprocessing utilized `Pandas` and `NumPy`. Data preprocessing occurred within this resource-slack environment.

2) Resource Constrained Environment: The Raspberry Pi Zero is a prevalent choice in RCE, enabling innovative solutions in applications like those explored by Gómez-Carmona et al. [21, 22]. To mimic this environment for the research, a virtual machine with Linux OS, a 1GHz single-core CPU, and 512MB RAM was used to match Raspberry Pi Zero specifications. Model development utilized Python 3.9.7, `Keras`, `TensorFlow`, and `Scikit-learn`.

B. Exploratory Data Analysis

Exploratory Data Analysis (EDA) is pivotal for AI model optimization. This study conducted EDA on a subset of the IoT-23 dataset.

1) EDA on IoT-23: The IoT-23 dataset consisted of 20 *Zeek* log files, each approximately 8GB in size. The `Zat` library was used to convert these logs into data frames. Statistical analysis helped identify and remove irrelevant and redundant features.

Addressing imbalanced datasets is crucial in ML, especially when one class greatly outweighs others, leading to biased model performance. To combat this, under-sampling techniques were applied to balance class distribution, improving performance on the minority class. However, care was taken to prevent information loss.

For streamlined analysis, the "label" and "detailed_label" features were merged post under-sampling, consolidating similar attack types. This enhanced workflow efficiency and simplicity. This merger streamlined data analysis, reducing redundancy and complexity. It enabled a concise representation of data and focused analysis on unified attack types.

Various techniques handled data quality concerns like missing and duplicate values. Normalization ensured consistent feature scales, preventing dominance by any single feature.

Class imbalance led to removing classes 3, 4, and 6, resulting in a balanced dataset of 63,000 instances (9,000 per class). An 80:20 split accommodated model training and

evaluation, upholding best practices for comprehensive model assessment and robust training.

TABLE I: Class Categories of Malware in the IoT-23 Dataset

Class	Categories of Malware	Total samples
0	Benign	30,860,691
1	C&C, C&C-FileDownload	22,048
2	C&C-HeartBeat, C&C-HeartBeat-Attack, C&C-HeartBeat-FileDownload	34,518
3	C&C-Mirai	2
4	C&C-Torii	30
5	DDoS	19,538,713
6	FileDownload	18
7	Okiru, Okiru Attack	609,907,11
8	PartOfAHorizonalPortScan, PartOfAHorizonalPortScan-Attack, C&CPartOfAHorizonalPortScan	213,853,817
9	Attack	9,398

C. Convolutional Neural Network

The model evaluated is a 1D-CNN architecture for feature extraction and classification. It begins with a convolutional layer for data insights featuring 32 filters to capture local patterns. The output tensor shape is (None, 10, 32), indicating variable batch size, sequence length of 10, and 32 filters.

A subsequent max pooling layer reduces dimensions while retaining vital features, leading to a tensor shape of (None, 5, 32). This enhances efficiency by preserving important details and discarding less relevant information.

Another convolutional layer, employing 64 filters, further enhances feature extraction for complex patterns. This reduces sequence length to 3 and yields a tensor shape of (None, 3, 64). Increased filters empower the model to learn intricate patterns, thus improving its classification capabilities.

Succeeding layers (flatten, dense with *sigmoid* activation, and output dense layer with *softmax* activation) proficiently classify the features into seven classes. These layers process and transform data, facilitating accurate predictions and effective classification of new instances.

TABLE II: Convolutional Neural Network Model Summary

Layer	Output Shape	Parameter #
conv1d (Conv1D)	(None, 10, 32)	128
max_pooling1d (MaxPooling1D)	(None, 5, 32)	0
conv1d_1(Conv1D)	(None, 3, 64)	6208
flatten (Flatten)	(None, 192)	0
dense (Dense)	(None, 128)	24704
dense_1 (Dense)	(None, 7)	1290
Total Parameters: 32,330		
Trainable Parameters: 32,330		
Non-Trainable Parameters: 0		

D. Dilated Convolutional Neural Network

One lightweight variant of a CNN is the 1D Dilated Convolutional Neural Network (1D-DCNN), which is notably more efficient compared to standard CNNs [23, 24]. This evaluation presents a sequential 1D-DCNN model architecture, as outlined in Table II. This architecture adeptly extracts features and conducts data classification in diverse domains. The performance of the model on complex real-world problems benefits significantly from hyperparameter optimization.

The model's foundation is a convolutional layer employing 32 filters, a kernel size of 1, and a dilation rate of 2. This captures local patterns and important data features. The dilation rate expands the receptive field, enabling the extraction of broader, contextual information. Subsequent layers build upon the learned representations established by this layer.

The next layer has 64 filters, a kernel size of 1, and a dilation rate of 8. This layer enhances the model's capacity to learn higher-level representations and complex features. The larger dilation rate enables the capture of long-range dependencies and informative patterns across a wider context. Together, these two convolutional layers constitute a potent feature extraction module.

For the ultimate classification task, a flattened layer reshapes the output into a one-dimensional tensor, ensuring smooth integration with ensuing dense layers. The final dense layer employs `softmax` activation, generating the model's output for multi-class predictions. Training employs the *categorical cross-entropy* loss function and the `Adam` optimizer, further heightening classification accuracy.

TABLE III: 1D Dilated Convolutional Neural Network Model Summary

Layer	Output Shape	Parameter #
conv1d (Conv1D)	(None, 12, 32)	64
conv1d_1 (Conv1D)	(None, 12, 64)	2,112
flatten (Flatten)	(None, 768)	0
dense (Dense)	(None, 7)	7,690
Total Parameters: 9,866		
Trainable Parameters: 9,866		
Non-Trainable Parameters: 0		

E. Recurrent Neural Network

1) Long Short-Term Memory: Backpropagation through time in Recurrent Neural Networks (RNNs) is computationally intensive and slow [25]. This challenge arises from propagating error signals across extended time intervals, where signals can either exponentially increase or diminish. The introduction of Long Short-Term Memory (LSTM) architecture, a specialized RNN variant, effectively addresses this issue. LSTMs can manage time lags exceeding 1,000 discrete steps, rendering them more efficient for training RNNs. The model structure in Table IV comprises an LSTM layer and a subsequent dense layer, designed to capture and learn temporal dependencies.

The LSTM layer tackles the vanishing gradient problem prevalent in conventional RNNs. Outputting a shape of (None, 64) and featuring 19,712 trainable parameters, LSTMs utilize memory cells and gating mechanisms to selectively retain or discard information across multiple time steps. This capability

facilitates capturing prolonged dependencies and sustaining contextual memory. Consequently, LSTMs excel in scenarios with temporal gaps, making them favored in Natural Language Processing (NLP), speech recognition, and time series forecasting due to their proficiency in deciphering complex temporal patterns.

Following the LSTM layer, a dense layer significantly contributes to the model's performance. With an output shape of (None, 10) and 650 parameters, this layer introduces non-linear transformations through activation functions, such as `softmax`. The dense layer abstracts learned features into higher-level representations, converting information into class probabilities. This mechanism enables predictions across various classes, supporting decision-making and interpretation. The simplicity and interpretability of the dense layer provide valuable insights into the model's reasoning process and facilitate the understanding of learned representations.

2) Gated Recurrent Unit: The GRU model is a versatile choice for sequential data analysis, offering unique benefits across applications. The core GRU layer, with (None, 64) output shape and 14,976 parameters, excels in capturing long-term dependencies. It surpasses traditional RNNs by using gating mechanisms to regulate information flow, adaptively retaining or discarding information from prior steps. The GRU's efficiency in learning temporal dependencies makes it valuable for various tasks like NLP, sentiment analysis, and music generation, handling sequences with diverse lengths and complex patterns.

Following the GRU layer, a dense layer improves model performance. With (None, 10) output shape and 650 parameters, it introduces non-linear transformations using activation functions like `softmax`. This layer maps GRU-learned features to class probabilities, aiding predictions. Its compact size ensures computational efficiency and scalability for real-time applications and large-scale datasets.

Table V summarizes the model architecture, including layer details, output shapes, and trainable parameters. All parameters are adjustable during training for task optimization.

TABLE IV: Long Short Term Memory Model Summary

Layer	**Output Shape**	**Parameter #**
lstm (LSTM)	(None, 64)	19,712
dense (Dense)	(None, 10)	650
Total Parameters: 20,362		
Trainable Parameters: 20,362		
Non-Trainable Parameters: 0		

TABLE V: Gated Recurrent Unit Model Summary

Layer	**Output Shape**	**Parameter #**
gru (GRU)	(None, 64)	14,976
dense_1 (Dense)	(None, 10)	650
Total Parameters: 15,626		
Trainable Parameters: 15,626		
Non-Trainable Parameters: 0		

F. Machine Learning Algorithms

For a comprehensive ML model selection, a meticulous experimental methodology was adopted. This involved training and evaluating widely-used algorithms like KNN, RF, DT, XGBoost, and NB. The `GridSearchCV` function from `Scikit-learn` library was employed for performance optimization. It exhaustively searched hyperparameters, aiming to identify configurations yielding high accuracy and generalization. The parameter search process entailed defining relevant hyperparameter values for each model. For instance, RF considered estimators, max features per split, and splitting criterion.

Post hyperparameter optimization, models were trained with selected settings. This phase facilitated learning data patterns, enabling accurate predictions on new instances.

V. Results

The experimental evaluation of both DL and ML models exhibited promising outcomes in terms of accuracy, CPU usage, and execution time on IoT-23 datasets. CPU usage assessment leveraged the `Psutil` library, which offers insights into processes, system utilization (CPU, memory, disks, network), aiding monitoring, profiling, and resource allocation. `Psutil` is versatile, compatible with various operating systems and architectures [26].

Key evaluation metrics encompassed accuracy, precision, recall, and F1 score, gauging model classification performance. These metrics collectively appraised the model's capacity to identify accurately, minimize misclassifications, capture positive instances, and maintain balanced precision and recall.

A. Result Analysis

In the evaluation phase, DL models were assessed by tracking accuracy and loss on training and validation sets at each epoch. This facilitated anomaly detection and accurate validation data identification. The loss function computed gradients, essential for updating biases and weights in neural networks. The *Adam* optimizer and *sparse categorically cross-entropy* loss function were employed.

To counter overfitting, an early stopping technique halted training when validation loss ceased to decrease after a set number of iterations. DL models underwent 15 epochs with a batch size of 32. Figure 1 depicts loss and accuracy's inverse relationship. Training set accuracy approximated 99%, while validation set accuracy slightly exceeded at 99.2%. Further epochs did not notably elevate accuracy due to potential overfitting.

Confusion matrices (Figure 4) offered classification insights on IoT-23. ML algorithms (Table VI) showed varying performance. DT and RF scored 99.99% accuracy, XGBoost reached 99.8%. DL models (1D-CNN, 1D-DCNN, LSTM, GRU) achieved accuracy from 98.5% to 99.2%, displaying competence in intricate pattern recognition.

CPU usage differed among models. RF consumed 40%, DT and NB used 8.26% and 11.05% respectively. XGBoost

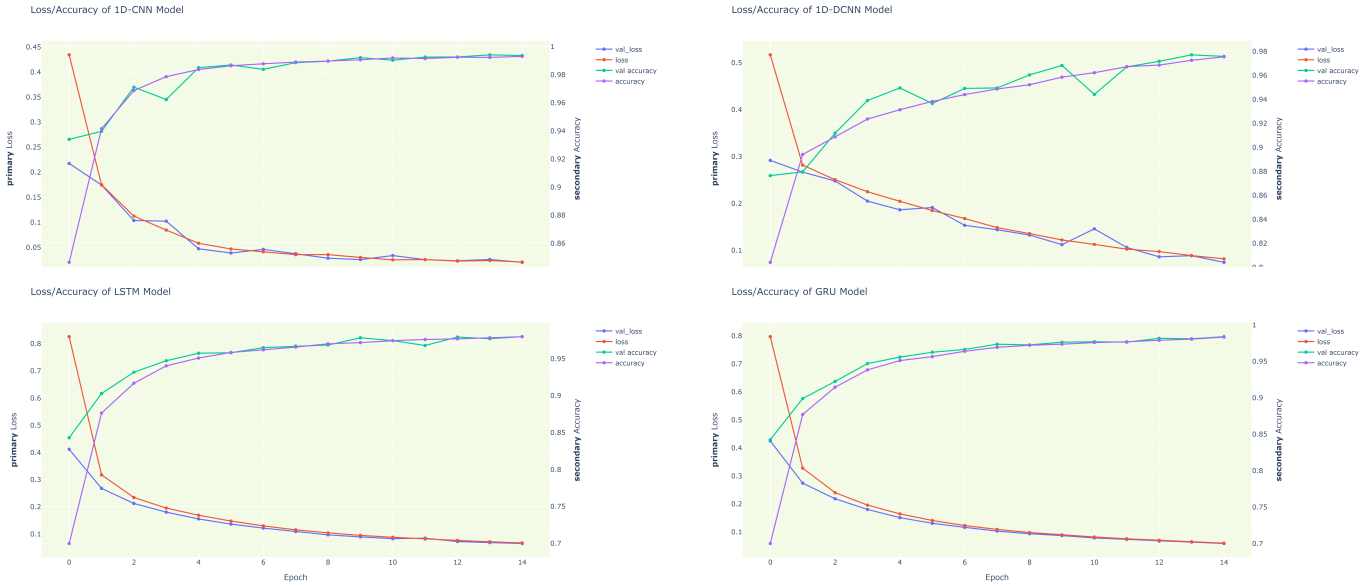


Fig. 1: DL Models Performance on the Training Dataset of IoT-23

TABLE VI: Evaluation of AI Models on the IoT-23 Dataset in the Slack and Resource-Constrained Environments

SNo	Model	PC Training time (s)	PC CPU Usage (%)	PI Training time (s)	PI CPU Usage (%)	Accuracy (%)	Precision	Recall	F1-Score
1	DT	0.046	1.13	1.332	8.26	99.99	0.999	0.999	0.999
2	RF	1.979	6.82	6.834	40.01	99.99	0.999	0.999	0.999
3	KNN	0.078	0.93	0.371	12.45	91.40	0.918	0.914	0.914
4	NB	0.001	0.61	0.020	11.05	54.10	0.595	0.541	0.509
5	XGBoost	3.19	48.21	587.52	74.34	99.98	0.998	0.998	0.998
6	1D-CNN	30.065	28.37	313.11	20.10	99.20	0.992	0.992	0.992
7	1D-DCNN	31.862	26.12	330.71	18.13	98.70	0.987	0.987	0.987
8	LSTM	35.555	31.04	449.36	28.07	98.65	0.98.6	0.98.5	0.98.6
9	GRU	36.395	23.85	472.56	22.11	98.55	0.985	0.985	0.985

utilized 57.8%. DL models (1D-CNN, 1D-DCNN, LSTM, GRU) employed 20% to 28%.

LSTM required the longest training (approximately 449 seconds) in the constrained environment. DT, RF, NB, and KNN trained faster than XGBoost and DL models.

TABLE VII: Accuracy Comparison of the DT and RF Models used in this Work against Existing Approaches

Reference	Dataset	Classification	Accuracy (%)
Hegde et al. [27]	IoT-23	Multi-class	99.90
ElKashlan et al. [28]	IoT-23	Multi-class	99.20
Vitorino et al. [29]	IoT-23	Multi-class	99.97
Kanimozhi and Jacob [30]	IoT-23	Multi-class	99.96
Wei et al. [31]	IoT-23	Multi-class	99.92
Evaluated DT and RF Models	**IoT-23**	**Multi-class**	**99.99**

Figure 3 and Figure 2 offer a comprehensive overview of model performance, including accuracy, training time, and CPU usage, in both slack and RCEs. These figures display performance discrepancies and trade-offs between accuracy and resource consumption. For further accuracy context, Table VII compares the proposed NIDS with other existing works.

The study reveals distinct features and trade-offs between ML and DL models. DT and RF excel in accuracy, while DL models achieve impressive rates above 99%, demanding more computational resources.

Overall, the results demonstrate the resource-constrained potential for DL and ML approaches to NIDSs, and highlight the effectiveness of the proposed AI models in detecting and categorizing network attacks. This facilitates the efficient storage of pertinent network traffic for future analysis, i.e., benign packets can be discarded at the edge itself. These findings have important implications for organizations seeking to improve their cybersecurity posture and provide a valuable contribution to the growing body of literature on intrusion detection using ML and DL approaches.

VI. Discussion

The integration of IoT and edge devices brings advantages but also potential security vulnerabilities, necessitating NIDS and forensics readiness for data protection and incident investigation [3]. A multiclassification NIDS improves network

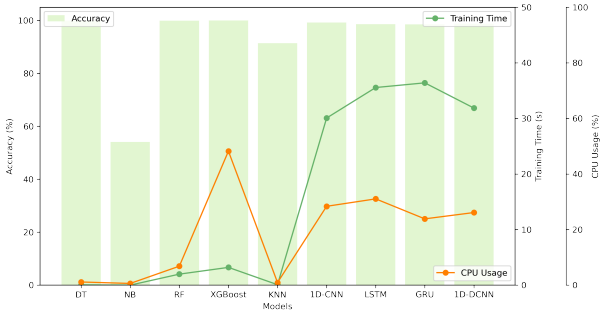


Fig. 2: Comparison of Proposed AI Models in Terms of Accuracy, Training Time, and CPU Usage on Slack Environment

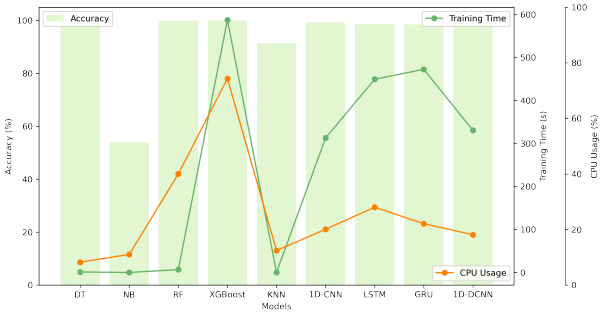


Fig. 3: Comparison of Proposed AI Models in Terms of Accuracy, Training Time, and CPU Usage on Constrained Environment

forensic readiness in resource-constrained settings, offering evidence and insights into security incidents involving edge devices. This approach enhances monitoring, detection, and security of edge networks.

The benefits of a multiclass NIDS in resource-constrained environments include:

- **Evidence Collection**: Gathers essential evidence while optimizing resource use.
- **Threat Identification**: Classifies security threats, aiding investigations.
- **Real-Time Monitoring**: Alerts to incidents in real time.
- **Automated Analysis**: Automates log and network data analysis for insights.
- **Efficient Investigations**: Centralizes evidence collection and analysis for quicker root cause identification.

Training lightweight DL models on Raspberry Pi Zero demonstrates potential in enhancing RCE security and forensic readiness. While direct training in RCEs faces limitations due to computational and memory constraints, validation can be done locally. Cloud infrastructure can be leveraged for training the models, and these pre-trained models can be deployed on edge devices – achieving the aforementioned benefits while eliminating many limitations.

A. Limitations

The proposed approach also has a number of limitations:

- The suggested NIDS approaches only investigated a single dataset, i.e., IoT-23. The performance of the proposed models is worth investigating on other IoT datasets, such as CICIoT2023 [32].
- The lower threshold of computational power needed to successfully run the developed models on resource-constrained physical devices has not yet been evaluated.
- The comparison of the execution of the proposed models using TensorFlow, TensorFlow-light and TinyML has not yet been evaluated.
- The assessment of power consumption on edge devices during the execution of the proposed models has not been thoroughly investigated. Understanding the power requirements and evaluating the energy efficiency of the models is crucial for resource-constrained environments.

VII. Conclusion

In response to security challenges in diverse IoT/edge networks, this paper evaluates AI-based network intrusion detection techniques in RCEs. It involves training lightweight classification algorithms directly on these devices, enhancing security and enabling real-time attack detection with minimal computational load. The evaluation, conducted on a low-powered device, showcased accuracy rates of over 99% on IoT-23 dataset across multiple attack types. This approach not only benefits academia and industry but also highlights the significance of considering factors like training time and CPU usage when selecting intrusion detection methods. By empowering edge devices with security tasks, this approach transforms traditional centralized security models and ensures swift responses to threats through edge-based security measures.

References

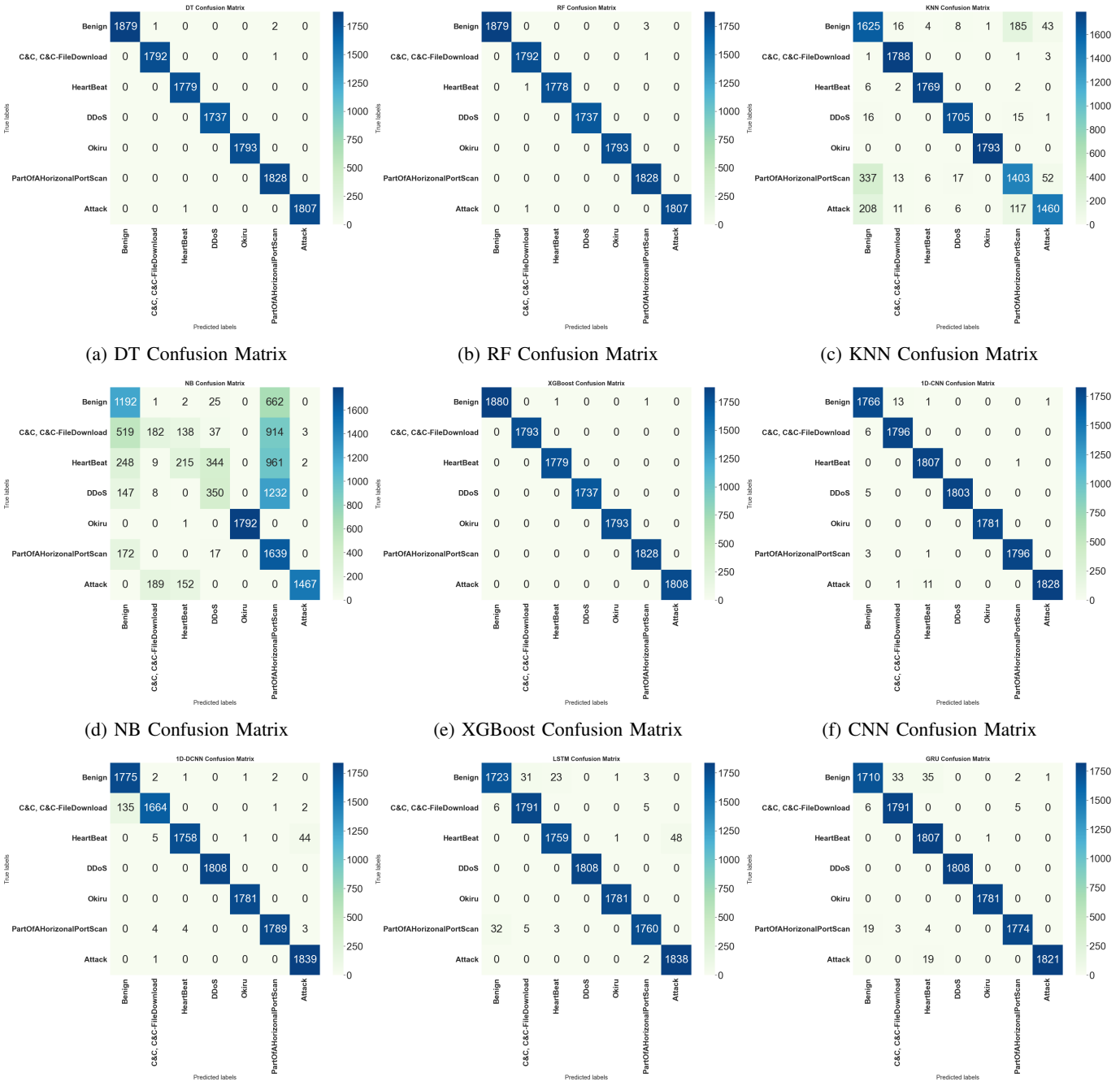
[1] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," *Ad Hoc Networks*, vol. 10, no. 7, pp. 1497–1516, 2012.

[2] E. van de Wiel, M. Scanlon, and N.-A. Le-Khac, "Enabling Non-Expert Analysis of Large Volumes of Intercepted Network Traffic," in *Advances in Digital Forensics XIV*, G. Peterson and S. Shenoi, Eds. Cham: Springer International Publishing, 2018, pp. 183–197.

[3] K. Friday, E. Bou-Harb, J. Crichigno, M. Scanlon, and N. Beebe, "Offloading network forensic analytics to programmable data plane switches," in *Innovations in Digital Forensics*. World Scientific, 2023, pp. 139–190.

[4] S. Sachintha, N.-A. Le-Khac, M. Scanlon, and A. P. Sayakkara, "Data exfiltration through electromagnetic covert channel of wired industrial control systems," *Applied Sciences*, vol. 13, no. 5, 2023.

[5] P. Hu, S. Dhelim, H. Ning, and T. Qiu, "Survey on fog computing: architecture, key technologies, applications and open issues," *Journal of Network and Computer Applications*, vol. 98, pp. 27–42, 2017.



(a) DT Confusion Matrix (b) RF Confusion Matrix (c) KNN Confusion Matrix

(d) NB Confusion Matrix (e) XGBoost Confusion Matrix (f) CNN Confusion Matrix

(g) 1D-DCNN Confusion Matrix (h) LSTM Confusion Matrix (i) GRU Confusion Matrix

Fig. 4: Confusion Matrix of Employed AI Models on IoT-23 Dataset

[6] J. Ren, D. Zhang, S. He, Y. Zhang, and T. Li, "A Survey on End-Edge-Cloud Orchestrated Network Computing Paradigms: Transparent Computing, Mobile Edge Computing, Fog Computing, and Cloudlet," *ACM Computing Surveys (CSUR)*, vol. 52, no. 6, pp. 1–36, 2019.

[7] S. Rizvi, M. Scanlon, J. McGibney, and J. Sheppard, "Application of Artificial Intelligence to Network Forensics: Survey, Challenges and Future Directions," *IEEE Access*, vol. 10, pp. 110 362–110 384, 2022.

[8] S. Garcia, A. Parmisano, and M. J. Erquiaga, "IoT-23: A labeled dataset with malicious and benign IoT network traffic," Jan. 2020, More details available at https://www.stratosphereips.org/datasets-iot23.

[9] X. Du, C. Hargreaves, J. Sheppard, F. Anda, A. Sayakkara, N.-A. Le-Khac, and M. Scanlon, "SoK: Exploring the State of the Art and the Future Potential of Artificial Intelligence in Digital Forensic Investigation," in *Proceedings of the 15th International Conference on Availability, Reliability and Security*, ser. ARES '20. New York, NY, USA: Association for

Computing Machinery, 2020.

[10] V. R. Kebande, P. P. Mudau, R. A. Ikuesan, H. Venter, and K.-K. R. Choo, "Holistic digital forensic readiness framework for IoT-enabled organizations," *Forensic Science International: Reports*, vol. 2, p. 100117, 2020.

[11] A. Valjarević, H. Venter, and R. Petrović, "ISO/IEC 27043:2015 - Role and application," in *2016 24th Telecommunications Forum (TELFOR)*. IEEE, 2016, pp. 1–4.

[12] A. S. Almogren, "Intrusion detection in Edge-of-Things computing," *Journal of Parallel and Distributed Computing*, vol. 137, pp. 259–265, 2020.

[13] I. Idrissi, M. Mostafa Azizi, and O. Moussaoui, "A Lightweight Optimized Deep Learning-based Host-Intrusion Detection System Deployed on the Edge for IoT," *International Journal of Computing and Digital System*, 2021.

[14] H. Hindy, C. Tachtatzis, R. Atkinson, E. Bayne, and X. Bellekens, "MQTT-IoT-IDS2020: MQTT-IoT-IDS2020 Internet of Things Intrusion Detection Dataset," *IEEE Dataport*, 2020.

[15] "TensorFlow Lite," https://www.tensorflow.org/lite, 2017, accessed on June 11, 2023.

[16] H. Djigal, J. Xu, L. Liu, and Y. Zhang, "Machine and deep learning for resource allocation in multi-access edge computing: A survey," *IEEE Communications Surveys & Tutorials*, 2022.

[17] S. S. Saha, S. S. Sandha, and M. Srivastava, "Machine learning for microcontroller-class hardware-a review," *IEEE Sensors Journal*, 2022.

[18] R. Kolcun, D. A. Popescu, V. Safronov, P. Yadav, A. M. Mandalari, Y. Xie, R. Mortier, and H. Haddadi, "The case for retraining of ML models for IoT device identification at the edge," *arXiv preprint arXiv:2011.08605*, 2020.

[19] K. Peffers, M. Rothenberger, T. Tuunanen, and R. Vaezi, "Design science research evaluation," in *Design Science Research in Information Systems. Advances in Theory and Practice*, K. Peffers, M. Rothenberger, and B. Kuechler, Eds. Springer, 2012, pp. 398–410.

[20] Stratosphere, "Stratosphere Laboratory Datasets," 2015, retrieved July 5, 2023, from https://www.stratosphereips.org/datasets-overview.

[21] O. Gómez-Carmona, D. Casado-Mansilla, D. López-de Ipiña, and J. García-Zubia, "Simplicity is Best: Addressing the Computational Cost of Machine Learning Classifiers in Constrained Edge Devices," in *Proceedings of the 9th International Conference on the Internet of Things*, 2019, pp. 1–8.

[22] ——, "Optimizing Computational Resources for Edge Intelligence Through Model Cascade Strategies," *IEEE Internet of Things Journal*, vol. 9, no. 10, pp. 7404–7417, 2021.

[23] S. Rizvi, M. Scanlon, J. McGibney, and J. Sheppard, "Deep Learning Based Network Intrusion Detection System for Resource Constrained Environments," in *Proceedings of the 13th EAI International Conference on Digital Forensics and Cyber Crime (ICDF2C)*. Springer, 2023, pp. 1–7.

[24] S. M. Rizvi, T. Syed, and J. Qureshi, "Real-time forecasting of petrol retail using dilated causal CNNs," *Journal of Ambient Intelligence and Humanized Computing*, pp. 1–12, 2022.

[25] J. Kim, H. Kim *et al.*, "An Effective Intrusion Detection Classifier Using Long Short-Term Memory with Gradient Descent Optimization," in *2017 International Conference on Platform Technology and Service (PlatCon)*. IEEE, 2017, pp. 1–6.

[26] Giampaolo Rodola'. (2023) Psutil. [Accessed: July 5, 2023]. [Online]. Available: https://pypi.python.org/pypi/psutil/

[27] M. Hegde, G. Kepnang, M. Al Mazroei, J. S. Chavis, and L. Watkins, "Identification of Botnet Activity in IoT Network Traffic Using Machine Learning," in *2020 International Conference on Intelligent Data Science Technologies and Applications (IDSTA)*. IEEE, 2020, pp. 21–27.

[28] M. ElKashlan, M. S. Elsayed, A. D. Jurcut, and M. Azer, "A Machine Learning-Based Intrusion Detection System for IoT Electric Vehicle Charging Stations (EVCSs)," *Electronics*, vol. 12, no. 4, p. 1044, 2023.

[29] J. Vitorino, I. Praça, and E. Maia, "Towards adversarial realism and robust learning for IoT intrusion detection and classification," *Annals of Telecommunications*, pp. 1–12, 2023.

[30] V. Kanimozhi and T. P. Jacob, "The Top Ten Artificial Intelligence-Deep Neural Networks for IoT Intrusion Detection System," *Wireless Personal Communications*, vol. 129, no. 2, pp. 1451–1470, 2023.

[31] Y. Z. Wei, M. Md-Arshad, A. A. Samad, and N. Ithnin, "Comparing Malware Attack Detection using Machine Learning Techniques in IoT Network Traffic," *International Journal of Innovative Computing*, vol. 13, no. 1, pp. 21–27, 2023.

[32] E. C. P. Neto, S. Dadkhah, R. Ferreira, A. Zohourian, R. Lu, and A. A. Ghorbani, "CICIoT2023: A Real-Time Dataset and Benchmark for Large-Scale Attacks in IoT Environment," *Sensors*, vol. 23, no. 13, 2023.