Anomaly-based Intrusion Detection System Using ESP32-WROOM-DA

Biagio Boi
University of Salerno
Fisciano, Italy
bboi@unisa.it

Franco Cirillo
University of Salerno
Fisciano, Italy
fracirillo@unisa.it

Marco De Santis
University of Salerno
Fisciano, Italy
mdesantis@unisa.it

Christian Esposito
University of Salerno
Fisciano, Italy
esposito@unisa.it

ABSTRACT

Internet of Things (IoT) devices are increasingly employed in monitoring and controlling both domestic and industrial infrastructures. However, security measures are often neglected due to the computational resource limitations of these devices. Despite numerous research initiatives aimed at developing Intrusion Detection Systems (IDS) for IoT, practical implementation-focused studies remain scarce. The goal of this research is to develop an anomaly-based IDS, or more precisely, a detection engine of an IDS, using a supervised approach with three different neural network models: Sequential Neural Network (SNN), Recurrent Neural Network (RNN), and Deep Recurrent Neural Network (DRNN). The objective is to determine whether it is feasible to create a high-performing IDS, characterized by high accuracy, while simultaneously maintaining low resource requirements— a critical aspect when deploying on microcontrollers with limited hardware capabilities. The IDS must be capable of performing multiclass classification to distinguish between normal packet flows, DoS attacks, Probe attacks, and also binary classifications. To achieve this, the IDS is first trained and then tested on the NSL-KDD dataset. Feature extraction is conducted using both the Random Forest algorithm and the Shap algorithm. According to the results presented in the final chapter, the most accurate IDS utilizes the SNN model trained with features determined by Shap in binary classification, achieving a precision level of 94.04%. This IDS, when deployed on the ESP32-WROOM-32 microcontroller, reports a minimum inference time of 0.226 ms, an average time of 3.198 ms, and a maximum time of 10.478 ms, requiring just over 8 KB of SRAM for installation.

CCS CONCEPTS

• **Security and privacy** → *Authentication*; *Privacy-preserving protocols*;

KEYWORDS

Intrusion Detection System, Internet of Things, Embedded System

ACM Reference Format:
Biagio Boi, Franco Cirillo, Marco De Santis, and Christian Esposito. 2024. Anomaly-based Intrusion Detection System Using ESP32-WROOM-DA. In *INTERNATIONAL CONVERENCE ON CONVERGENT AND SMART SYSTEMS*. ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

The advent of the Internet of Things (IoT) has revolutionized the way information is shared and managed, creating an interconnected ecosystem of smart devices that permeates many aspects of daily life [8]. While this pervasive interconnection offers significant benefits in terms of efficiency and automation, it also poses new challenges, particularly with regard to data security and the protection of IoT networks [6]. Large-scale information sharing, made possible by the rapid growth of the Internet, has led to an exponential increase in data traffic. Today, the network generates an enormous amount of data, estimated at around 5 quintillion bytes per day [11]. This continuous flow of information poses a significant privacy and security risk, especially considering that the data in question often includes sensitive customer or business-related information, leading to possible threats by malicious nodes. The concrete motivation for the lack of security measures on these devices, as opposed to traditional servers and workstations, is their limited computing and energy resources.

In response to these challenges, Intrusion Detection System (IDS) plays a key role in the detection and response of cyber threats. In particular, anomaly-based IDS focuses on detecting abnormal behavior w.r.t. normal system operation, offering a promising perspective for detecting malicious and unusual activity within IoT environments.

Among all microcontrollers currently available in the IoT scenario, ESP32-WROOM-DA is one of the preferred ones when taking into account performance and security. This microcontroller, known for its low power consumption capabilities and flexibility, offers a solid basis for the development of a decision engine for an IoT-compatible IDS. In this context, this paper proposes the implementation of an anomaly-based IDS using the ESP32-WROOM-DA, with the aim of developing a system capable of automatically detecting suspicious behavior and intrusions in IoT networks. The development process includes creating meaningful data models for the IoT context, pre-processing the data, selecting relevant features, and implementing classification techniques to identify and promptly respond to threats. Given the increasing number of vulnerabilities in the IoT landscape, our research offers significant contributions:

- We evaluated three models, including Recurrent Neural Network (RNN) and Sequential Neural Network (SNN), within the context of anomaly-based Intrusion Detection Systems (IDS). Our performance analysis reveals important insights regarding the data used for classifying instructions.
- We assessed these models using a well-known and comprehensive dataset, such as NSL-KDD [12], thereby broadening the application scenarios of anomaly-based IDS beyond their previous limitation to MQTT communication [4][9].
- By combining the insights from the evaluation of the models and the expanded application scenarios, our research demonstrates the versatility and effectiveness of anomaly-based IDS in addressing a broader range of IoT vulnerabilities.

The paper is structured into six sections. In the second section, we will introduce related works in the field of IDS for the IoT context. In the third section, we will discuss the proposed architecture for anomaly-based IDS for the IoT, while in the fourth, we will motivate the success of our research by discussing results and making some considerations of the effectiveness and performance of the implemented system. We conclude our paper with the conclusion and future works.

2 RELATED WORK

Recent efforts in the domain of IDS targeting IoT devices have demonstrated a shift toward the deployment of these systems in constrained environments. These approaches aim to address specific challenges related to energy efficiency, inference performance, and model size, which are typical parameters to take into account when designing an IoT system.

S. Hosseininoorbin et al. [3] investigated the use of convolutional neural networks (CNNs) for NIDS on edge devices of the IoT. Their study evaluated inference times and energy efficiency using the ToN-IoT dataset, showcasing trade-offs between model complexity and device performance. Results indicated that deeper CNN architectures led to increased model size and energy consumption, highlighting the need for optimized models for IoT deployment.

Idriss Idrissi et al. [4] explored the feasibility of CNN-based IDS on low-power IoT devices like ESP32-WROOM-32. By leveraging TensorFlow Lite for model conversion, they achieved high detection accuracies while maintaining a compact model size suitable for deployment on edge devices. The study demonstrated practical implementations of CNN-based NIDS on various IoT hardware platforms. Liam Daly Manocchio et al. [9] took a different approach by employing machine learning techniques such as decision trees for intrusion detection on MQTT-based IoT networks. Their work focused on minimizing model size and achieving high detection rates, emphasizing efficient utilization of limited resources on IoT devices.

A recent study [1] addresses increasing cybersecurity attacks on IoT systems by proposing an Intrusion Detection System to improve security against DoS attacks using anomaly detection and ML techniques. The study highlights IoT's vulnerability due to its self-configuring and open nature, making it susceptible to both insider and outsider attacks. The IDS employs four supervised classifier algorithms—Decision Tree (DT), Random Forest (RF), K Nearest Neighbor (kNN), and Support Vector Machine (SVM)—and utilizes two feature selection algorithms, the Correlation-based Feature Selection (CFS) algorithm and the Genetic Algorithm (GA). The IoTID20 dataset was used for training, with DT and RF classifiers showing the best performance, particularly with features selected by GA. However, this study does not use neural networks, which are among the most powerful tools in the field of machine learning for IDS.

In summary, recent studies have showcased various methods for implementing IDS on IoT devices, leveraging machine learning and deep learning techniques. These efforts highlight the importance of optimizing models for specific hardware constraints while maintaining robust intrusion detection capabilities. Our work aims to build on these approaches, focusing on improving the efficiency and performance of IDS on IoT devices.

3 METHODOLOGY

In this section, we present the methodology employed in our research, focusing on the utilization of advanced neural network models tailored for the specific task of anomaly-based intrusion detection in IoT environments.

The main purpose of our study is to develop an advanced intrusion detection system based on neural networks for the IoT environment. Our approach is based on the use of three main neural network architectures adapted for the specific purpose:

Sequential Neural Network (SNN): The SNN is a feedforward neural network designed to process sequential data of fixed size efficiently. This model is composed of densely connected layers with specific activation functions. We use a three-layer architecture, with the first layer consisting of a dense layer of 41 neurons with ReLU activation, the second layer of a dense layer of 20 neurons also with ReLU activation, and the third layer consisting of a dense layer with a number of neurons corresponding to the output classes, using softmax activation for multiclass classification. The SNN is implemented using Keras, an API for high-level neural networks. We chose the SNN because of its ability to process sequential data of fixed size efficiently. This model is suitable for our application because it can handle incoming network traffic data in a structured manner, exploiting densely connected layers for final classification.

Recurrent Neural Network (RNN): The RNN is designed to process sequential data while maintaining an internal state. This model is particularly suitable for time series or sequence-based task analysis [13]. Our RNN architecture includes a Long Short-Term Memory (LSTM) layer with 41 nodes and ReLU activation, a second LSTM layer with 20 nodes and 20% dropout to prevent overfitting, and an output layer with appropriate activation based on classification type (e.g., softmax for multiclass classification). The RNN was chosen for its ability to maintain an internal state, making it ideal for analyzing time series or sequential data. This model is particularly suitable for capturing temporal dependencies in network traffic, using LSTM layers to handle input sequences.

Deep Recurrent Neural Network (DRNN): The DRNN extends the RNN architecture by incorporating multiple layers of LSTM units to capture complex temporal patterns. Our DRNN architecture includes two LSTM layers, the first with 41 nodes and the second with 20 nodes, both with dropout regularization to mitigate overfitting. The output layer is designed for binary or multiclass classification tasks. The DRNN extends the RNN architecture by incorporating multiple LSTM layers to capture complex temporal patterns. We opted for the DRNN to explore whether the introduction of multiple LSTM layers can improve anomaly detection performance in the context of the IoT environment.

3.1 Dataset and Preprocessing

The dataset used to train and test the models is NSL-KDD [12], which is a successor to the KDD99 dataset developed by the University of New Brunswick as part of the International Knowledge Discovery and Data Mining Tools Competition in 1999. The dataset features are listed in Table 2. NSL-KDD is designed as a cleaned-up version of KDD99, aimed at providing a more balanced and suitable dataset for intrusion detection system (IDS) development.

NSL-KDD comprises four subsets of data: KDDTrain+, KDDTest+, KDDTrain+_20Percent, and KDDTest-21. For this study, we primarily focus on the first two subsets, KDDTrain+ (training set) and KDDTest+ (test set).

Table 1: Total rows and rows per attack type in NSL-KDD dataset

Dataset	Total	Normal	DoS	Probe	U2R	R2L
KDDTrain+	125,697	67,343	45,927	11,656	52	995
KDDTest+	22,544	9,711	7,458	2,421	200	2,654

The total number of rows in the NSL-KDD dataset is over 148,000, with a detailed distribution for each attack type. To ensure a balanced analysis, the User-to-Root (U2R) and Remote-to-Local (R2L) attack types are excluded from the analysis due to their limited number of samples.

The choice of the NSL-KDD dataset is motivated by its breadth and widespread use in the scientific literature for anomaly-based intrusion detection system development. Additionally, the dataset's sample size allows for reliable classification. Table 3 shows the precision of various works using the NSL-KDD dataset, highlighting its effectiveness in research contexts.

Table 3 demonstrates the precision achieved by various authors in the context of intrusion detection systems using the NSL-KDD dataset. These results underscore the dataset's effectiveness in cybersecurity applications.

For training and evaluation, we utilized the NSL-KDD dataset, which includes 42 features per record. This dataset was chosen due to its suitability for anomaly detection tasks in network traffic analysis. Our preprocessing steps involved:

- Feature selection to identify relevant input attributes.
- Normalization and scaling of input data.
- Partitioning the dataset into training and testing sets.

Table 2: Features of the NSL-KDD dataset

No.	Feature Name	No.	Feature Name
1	duration	22	is_guest_login
2	protocol type	23	count
3	service	24	srv_count
4	flag	25	serror_rate
5	src_bytes	26	srv_serror_rate
6	dst_bytes	27	rerror_rate
7	land	28	srv_rerror_rate
8	wrong_fragment	29	same_srvrate
9	urgent	30	diff_srv_rate
10	hot	31	srv_diff_host_rate
11	num_failed_logins	32	dst_host_count
12	logged_in	33	dst_host_srv_count
13	num_compromised	34	dst_host_same_srv_rate
14	root_shell	35	dst_host_diff_srv_rate
15	su_attempted	36	dst_host_same_src_port_rate
16	num_root	37	dst_host_srv_diff_host_rate
17	num_file_creations	38	dst_host_serror_rate
18	num_shells	39	dst_host_srv_serror_rate
19	num_access_files	40	dst_host_rerror_rate
20	num_outbound_cmds	41	dst_host_srv_rerror_rate
21	is_host_login	42	label

Table 3: Precision of IDS works using the NSL-KDD dataset in literature

Authors and References	Publication Year	Precision
Chuan-long et al. [15]	2017	83.49%
Jia et al. [5]	2017	83.58%
Abu Taher et al. [14]	2019	95.00%
Meliboev et al. [10]	2022	98.70%
Kasongo [7]	2023	88.13%
Chakrawarti et al. [2]	2023	99.95%

By leveraging these neural network architectures and preprocessing techniques on the NSL-KDD dataset, we aimed to develop an effective intrusion detection system for IoT environments.

The dataset underwent several preprocessing steps to prepare it for training neural network models. Initially, the dataset contained five attack classes, but the last two classes (U2R and R2L) were excluded due to insufficient samples, which could lead to imbalance and overfitting issues. Additionally, specific features (*protocol_type*, *service*, and *flag*) and labels were encoded into numerical values to make them compatible with the neural network models used for training.

For feature selection, the Random Forest algorithm and Shap (Shapley Additive Explanations) were employed to identify the most informative features for training the models. The Random Forest algorithm was used to assess feature importance, and the top 15 features were selected based on their significance. Similarly, Shap was utilized to extract important features, offering insights into the impact of each feature on model performance.

The Random Forest algorithm evaluated feature importance, resulting in the selection of the top 15 features based on their significance. These features were considered crucial for effective model training and classification.

Furthermore, Shap was used to identify and extract important features, providing detailed insights into each feature's impact on model performance.

The tables below summarize the most important features identified by the Random Forest and Shap methods for each neural network model, providing valuable insights into the dataset's key characteristics for effective model training and intrusion detection.

Table 4: Random Forest Feature Importance

No.	Feature Name
1	src_bytes
2	same_srv_rate
3	flag
4	diff_srv_rate
5	dst_bytes
6	count
7	dst_host_serror_rate
8	dst_host_diff_srv_rate

Table 5: Shap Feature Importance for SNN

No.	Feature Name
1	dst_host_rerror_rate
2	src_bytes
3	dst_host_srv_count
4	flag
5	logged_in
6	dst_host_same_srv_rate
7	same_srv_rate
8	same_srv_rate

Table 6: Shap Feature Importance for RNN

No.	Feature Name
1	dst_bytes
2	same_srv_rate
3	logged_in
4	dst_host_rerror_rate
5	dst_host_srv_rerrorrate
6	src_bytes
7	dst_host_same_src_port_rate
8	dst_host_same_srv_rate

These tables summarize the most important features identified by the Random Forest and Shap methods for each neural network model, providing valuable insights into the dataset's key characteristics for effective model training and intrusion detection.

The classification phase represents the culmination of model development, allowing us to evaluate the efficacy of our neural network models in both binary and multiclass scenarios. This work adopts a versatile approach, accommodating both types of classification tasks through appropriate model adaptations. Figure 1 illustrates the general architecture of our neural network models used for these classification tasks.

Table 7: Shap Feature Importance for DRNN

No.	Feature Name
1	dst_bytes
2	dst_host_rerror_rate
3	src_bytes
4	service
5	same_srv_rate
6	logged_in
7	dst_host_srv_serror_rate
8	dst_host_srv_rerror_rate

For binary classification, our models are tailored to utilize a sigmoid activation function in the output layer. The sigmoid function is defined as:

$$\sigma(u) = \frac{1}{1 + e^{-u}}$$

This activation function compresses the output values between 0 and 1, providing a clear indication of the likelihood of an instance belonging to a specific class. Values closer to 1 represent a confident prediction of normalcy, while values closer to 0 denote an anomaly.

In contrast, our models are equipped with a softmax activation function in the output layer for multiclass classification tasks. The softmax function transforms the model's raw outputs into a probability distribution across multiple classes. It is expressed as:

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

where K represents the total number of classes. The predicted class is determined by selecting the index with the highest probability value (argmax($\sigma(z)$)).

The subsequent phase involves converting our trained neural network models into a format suitable for deployment on the ESP32-WROOM-DA microcontroller. This conversion process is facilitated by the `convert_model` method provided by the EveryWhereML package. The resulting model is transformed into a header file (`snn_model.h`) optimized for microcontroller deployment.

It's important to note that only Sequential Neural Network (SNN) models leveraging essential SHAP (Shapley Additive Explanations) features are deployed onto the microcontroller. These models are selected based on their superior accuracy and computational efficiency, making them ideal for resource-constrained environments.

This comprehensive approach ensures that our models are not only effective in classification accuracy but also optimized for real-world deployment scenarios, particularly on microcontroller platforms with limited computational resources.

4 RESULTS

The models were initially evaluated using the following metrics:

- **Accuracy**: Measures the fraction of correct predictions out of the total number of predictions made by the model. Accuracy provides a general measure of overall model performance.

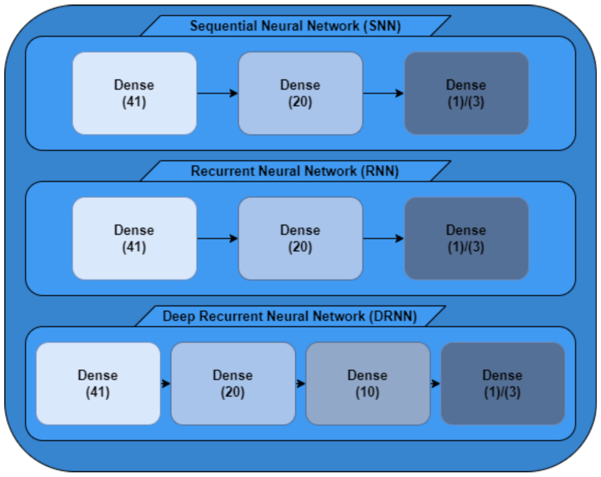

Figure 1: General structure of neural network models used in binary and multiclass classification tasks.

- **Positive Predictive Value (PPV)**: Measures the proportion of correct positive predictions out of the total number of positive predictions made by the model. PPV assesses the reliability of positive predictions.
- **Negative Predictive Value (NPV)**: Measures the proportion of correct negative predictions out of the total number of negative predictions made by the model. NPV assesses the reliability of negative predictions.

The inclusion of PPV and NPV metrics is context-dependent. Beyond overall model performance, understanding when the model performs best or worst is crucial, especially in intrusion detection scenarios. We provide confusion matrices for the best and worst results to visually identify model strengths and weaknesses.

4.1 Experiments

This section summarizes the experiments conducted with the developed models:

- **OnlyDoSProbe**: Includes samples from NSL-KDD dataset categorized as "normal", "dos", and "probe". All 41 available features are considered.
- **RandomForestIF**: Considers the top 15 important features from the NSL-KDD dataset determined by Random Forest.
- **SHAPSNNIF**: Utilizes the top 15 impactful features determined by SHAP (Sequential Neural Network).
- **SHAPRNNIF**: Uses the top 15 impactful features determined by SHAP (Recurrent Neural Network).
- **SHAPDRNNIF**: Incorporates the top 15 impactful features determined by SHAP (Deep Recurrent Neural Network).

4.2 Binary Classification Results

As shown in Table 8, the best-performing model is the SNN utilizing SHAP-derived features, while the worst-performing model is the DRNN using Random Forest-derived features. Across all models, there is a common trend of high PPV values, indicating strong precision for normal traffic predictions, but more challenges in predicting anomalies, particularly evident in models with LSTM layers, as reflected in the NPV values.

Table 8: Binary classification results of all models with different experiments

Model	Experiment	Accuracy (%)	PPV (%)	NPV (%)
SNN	OnlyDoSProbe	91.74	96.45	87.09
SNN	RandomForestIF	89.83	96.18	83.57
SNN	SHAPSNNIF	94.04	96.27	91.84
RNN	OnlyDoSProbe	85.48	98.03	73.14
RNN	RandomForestIF	83.35	96.27	70.65
RNN	SHAPRNNIF	89.74	98.41	81.20
DRNN	OnlyDoSProbe	83.29	98.64	68.20
DRNN	RandomForestIF	82.68	96.76	68.83
DRNN	SHAPDRNNIF	86.52	97.31	75.91

4.3 Multiclass Classification Results

Table 9: Multiclass classification results of all models with different experiments

Model	Experiment	Accuracy (%)	PPV (%)	NPV (%)
SNN	OnlyDoSProbe	88.89	96.68	83.72
SNN	RandomForestIF	88.22	96.94	81.36
SNN	SHAPSNNIF	89.82	97.20	84.00
RNN	OnlyDoSProbe	84.88	98.14	74.13
RNN	RandomForestIF	83.49	98.30	72.02
RNN	SHAPRNNIF	67.65	97.50	61.69
DRNN	OnlyDoSProbe	82.31	97.49	71.26
DRNN	RandomForestIF	79.78	97.53	65.58
DRNN	SHAPDRNNIF	66.08	96.70	59.59

Similarly, in Table 9, the best-performing model is the SNN with SHAP-derived features, while the worst-performing model changes to RNN with SHAP-derived features. Multiclass classification and confusion matrices confirm previous observations, with models struggling primarily in distinguishing between DoS and Probe attacks. Balancing sample differences could alleviate this issue.

4.4 Results on ESP32-WROOM-32

As mentioned earlier, the two most performant models in binary and multiclass classification were converted and subsequently deployed on the ESP32-WROOM-32 microcontroller. This microcontroller features:

- Processor: Xtensa dual-core 32-bit LX6, 240MHz.
- Memory: 520KB SRAM and 4MB Flash Memory.

Evaluation was conducted in terms of inference time and model resources. The results in terms of accuracy are omitted as they are equivalent to the previously discussed metrics. The evaluation was performed using 100 samples from the NSL-KDD dataset.

The two models exhibit similar inference times and model sizes. The binary classification model demonstrates lower minimum and maximum inference times compared to the multiclass classification model, while the binary classification model has a lower average

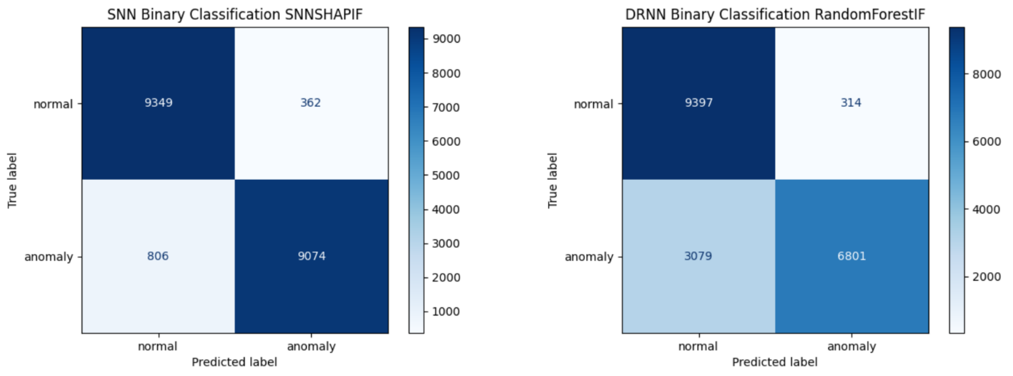

Figure 2: Confusion matrices for best and worst binary classification results

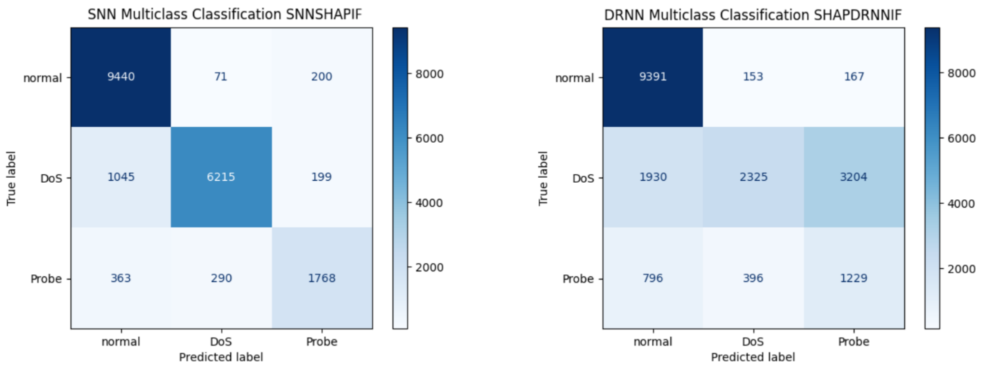

Figure 3: Confusion matrices for best and worst multiclass classification results

Table 10: Results of classification on ESP32-WROOM-32 microcontroller

Model	Experiment	TI Min	TI Med	TI Max	Model in Bytes
SNN	SHAPSNNIF Binary	0.226ms	3.198ms	10.478ms	8244B
SNN	SHAPSNNIF Multiclass	0.238ms	3.107ms	10.498ms	8328B

inference time. Table 11 provides a comparison of obtained results with related literature works involving IoT hardware testing.

4.5 Discussion

The binary classification results reveal that the SNN model utilizing SHAP-derived features consistently outperforms other models in terms of accuracy and positive predictive value (PPV). This suggests that the selected features identified by SHAP are highly effective for distinguishing normal traffic from anomalies. However, the lower negative predictive value (NPV) across models indicates an imbalance in the dataset considered.

The confusion matrices visually depict the model behavior, with the best-performing model showing strong performance in correctly classifying normal instances but struggling with anomaly detection, particularly evident in false negative cases.

In multiclass classification, the SNN model with SHAP-derived features again emerges as the top performer. Despite challenges in distinguishing between specific attack types like DoS and Probe, this model demonstrates superior accuracy and PPV compared to others.

The confusion matrices illustrate the difficulty in distinguishing between certain attack categories, suggesting potential areas for improvement such as dataset augmentation or model adjustments.

The deployment of models on the ESP32-WROOM-32 microcontroller highlights practical considerations for IoT applications. Both binary and multiclass models exhibit comparable inference times and model sizes, demonstrating feasibility for real-time inference on resource-constrained devices.

Comparisons with literature models underscore the competitive performance of our models on microcontroller hardware. Despite limitations in memory and processing power, our models achieve

Table 11: Comparison of the models developed with those present in the literature and installed on microcontrollers

Ref	Model	Classification	Dataset	TI Min	Device	Precision (%)
Proposed	SNN (8244B)	Binary	NSL-KDD	0,226ms	ESP32-WROOM-32	94.04
Proposed	SNN (8328B)	Multiclass	NSL-KDD	0,238ms	ESP32-WROOM-32	89.82
[3]	CNN (6000KB)	Multiclass	ToN	1ms	Google's Edge TPU	98.4
[3]	CNN (6000KB)	Multiclass	ToN	25ms	ARM Cortex-A53	98.4
[4]	CNN (106KB)	Multiclass	MQTT	1µs	Cortex-A53	99.74
[4]	CNN (4237B)	Multiclass	MQTT	2µs	ESP32-WROOM-32	97.2
[9]	D. Tree (8670B)	Multiclass	MQTT	1µs	ESP32-WROOM-32	99.92

comparable accuracy to larger, specialized hardware like Google's Edge TPU. Overall, the proposed model has been evaluated on a larger dataset with respect to [4] [9], which propose their work on a dataset of MQTT-related threats. We acknowledge the limitation in the accuracy of the proposed algorithm but we also want to highlight the performance over multiple classes of the models.

5 CONCLUSIONS

In this research work, we have developed a decision engine for an anomaly-based IDS using deep learning models, specifically feed-forward and feedback neural networks. From our experiments, the feed-forward models demonstrated superior performance compared to the feedback approaches. While the achieved precision levels are significant, it is worth noting that more advanced approaches exist in the scientific literature.

The proposed methodology provides guidelines on how to prepare a robust and deployable model in a real IoT environment. However, it is crucial to customize the IDS for the specific architecture of the IoT device in which it will be implemented. Each IoT device has unique hardware characteristics and resource constraints, necessitating a targeted approach in IDS design.

We evaluated the performance of our models using samples from the NSL-KDD dataset; however, conducting more realistic tests in actual IoT environments would be desirable. This would involve implementing the entire architecture described in the Background section, including the necessary hardware and software components for IDS operation.

Furthermore, looking ahead to sustained use in real-world scenarios, implementing a targeted reinforcement learning approach could be beneficial. This would enable the IDS to dynamically adapt to environmental variations and support extended periods without the need for frequent model updates.

In conclusion, the work undertaken provides a solid foundation for developing anomaly-based IDSs for IoT environments, highlighting future challenges and opportunities to enhance the effectiveness and adaptability of such systems. The goal remains to create robust cybersecurity solutions optimized for the complex and dynamic nature of IoT networks.

ACKNOWLEDGEMENTS

This work was supported by the Korea Institute of Energy Technology Evaluation and Planning (KETEP) grant funded by the Korea government (MOTIE) (RS-2023-00303559, Study on developing cyber-physical attack response system and security management system to maximize real-time distributed resource availability).

REFERENCES

[1] Esra Altulaihan, Mohammed Amin Almaiah, and Ahmed Aljughaiman. 2024. Anomaly Detection IDS for Detecting DoS Attacks in IoT Networks Based on Machine Learning Algorithms. *Sensors* 24, 2 (2024). https://doi.org/10.3390/s24020713
[2] Ankit Chakrawarti and Shiv Shakti Shrivastava. 2023. Intrusion Detection System using Long Short-Term Memory and Fully Connected Neural Network on Kddcup99 and NSL-KDD Dataset. *International Journal of Intelligent Systems and Applications in Engineering* 11, 9s (2023), 621–635.
[3] Seyedehfaezeh Hosseininoorbin, Siamak Layeghy, Mohanad Sarhan, Raja Jurdak, and Marius Portmann. 2023. Exploring edge TPU for network intrusion detection in IoT. *J. Parallel and Distrib. Comput.* 179 (2023), 104712.
[4] Idriss Idrissi, Mostafa Mostafa Azizi, and Omar Moussaoui. 2021. A lightweight optimized deep learning-based host-intrusion detection system deployed on the edge for IoT. *International Journal of Computing and Digital System* (2021).
[5] Yang Jia, Meng Wang, and Yagang Wang. 2019. Network intrusion detection algorithm based on deep neural network. *IET Information Security* 13, 1 (2019), 48–53.
[6] Ashwin Karale. 2021. The challenges of IoT addressing security, ethics, privacy, and laws. *Internet of Things* 15 (2021), 100420.
[7] Sydney Mambwe Kasongo. 2023. A deep learning technique for intrusion detection system using a Recurrent Neural Networks based framework. *Computer Communications* 199 (2023), 113–125.
[8] Asif Ali Laghari, Kaishan Wu, Rashid Ali Laghari, Mureed Ali, and Abdullah Ayub Khan. 2021. A review and state of art of Internet of Things (IoT). *Archives of Computational Methods in Engineering* (2021), 1–19.
[9] Liam Daly Manocchio, Siamak Layeghy, and Marius Portmann. 2022. Network intrusion detection system in a light bulb. In *2022 32nd International Telecommunication Networks and Applications Conference (ITNAC)*. IEEE, 1–8.
[10] Azizjon Meliboev, Jumabek Alikhanov, and Wooseong Kim. 2022. Performance evaluation of deep learning based network intrusion detection system across multiple balanced and imbalanced datasets. *Electronics* 11, 4 (2022), 515.
[11] Kuldeep Nagi. 2020. From bits and bytes to big data-An historical overview. *Available at SSRN 3622921* (2020).
[12] Danijela D Protić. 2018. Review of KDD Cup '99, NSL-KDD and Kyoto 2006+ datasets. *Vojnotehnički glasnik/Military Technical Courier* 66, 3 (2018), 580–596.
[13] Hojjat Salehinejad, Sharan Sankar, Joseph Barfett, Errol Colak, and Shahrokh Valaee. 2017. Recent advances in recurrent neural networks. *arXiv preprint arXiv:1801.01078* (2017).
[14] Kazi Abu Taher, Billal Mohammed Yasin Jisan, and Md Mahbubur Rahman. 2019. Network intrusion detection using supervised machine learning technique with feature selection. In *2019 International conference on robotics, electrical and signal processing techniques (ICREST)*. IEEE, 643–646.
[15] Chuanlong Yin, Yuefei Zhu, Jinlong Fei, and Xinzheng He. 2017. A deep learning approach for intrusion detection using recurrent neural networks. *Ieee Access* 5 (2017), 21954–21961.