Securing IoT Servers: Strategies for Employing Shallow and Deep Neural Networks

Niranjan W. Meegammana*[1], Harinda Fernando[2]
[1]Shilpa Sayura Foundation, [2]Sri Lanka Institute of Information Technology, Sri Lanka

***Abstract*—This study investigates the potential employability of Shallow and Deep Feed Forward Neural Networks (FFNs) in detecting attacks on low-resourced IoT application servers. It employed a Shallow FFN model with a single hidden layer of 512 neurons, and a Deep FFN model with 7 hidden layers, having between 256 to 4 neurons respectively. The study constructed four Shallow and Deep FFN models, utilizing two balanced UNSW-NB15 datasets containing 20 and 40 features. Experiments were conducted to detect network attacks on IoT networks. The results demonstrated that the Deep FFN model utilizing 40 features, despite slightly longer prediction times and higher resource usage, consistently outperformed other models, achieving an accuracy of 98.37%. Therefore, Deep FFN models prove suitable for protecting high-resourced IoT application servers. The Shallow model, achieving a faster detection time and moderate accuracy of 93%, is potentially employable in resource-constrained, low-latency IoT servers. This research enhances IoT security by employing Shallow and Deep FFN models based on different resource levels in IoT environments. Furthermore, it proposes integrating the Deep model into next-generation firewall systems to protect higher-value IoT servers. Future work involves exploring hybrid FFN architectures for protecting edge servers from network attacks.**

***Index Terms*—IoT Server Security, Neural Networks, Network Attack Detection, Deep and Shallow models**

I. Introduction

Application servers deployed as central nodes in IoT (Internet of Things) environments, such as healthcare facilities and smart cities, play a pivotal role in delivering critical services in the Industry 4.0 Digital Economy [1]. The rapid proliferation of IoT systems for data aggregation and communication has created a massive attack surface, making them prime targets for malicious actors. The inherent vulnerabilities in IoT, caused by the heterogeneity of devices and platforms, resource constraints, lack of security standards, and physical exposure, allow threat actors to exploit IoT networks. These vulnerabilities enable unauthorized access, service disruptions, theft of sensitive information, device abuse, and large-scale DDoS attacks [2]. Such attacks, capable of disrupting critical services, pose significant r isks t o b usinesses, a nd national economies, and potentially harm lives. Conventional security measures struggle to mitigate sophisticated cyber attacks targeting low-resource IoT infrastructure, which differs from traditional network architectures due to increased exposure, sheer distribution, and real-time low latency nature. Therefore, low-resourced IoT application servers demand robust security measures against escalating network threats across various sectors [3]. Figure 1 illustrates an IoT environment.

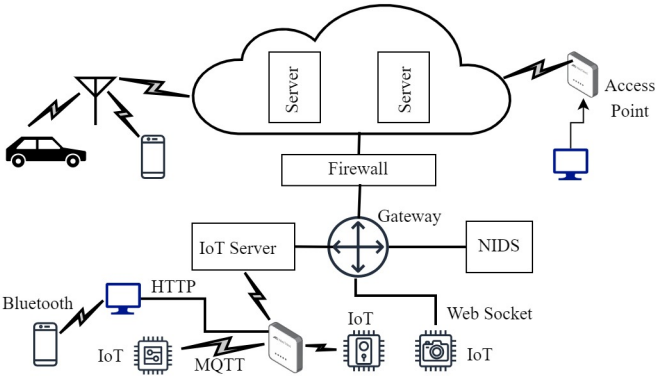

Fig. 1: An IoT environment

Deep Learning (DL) offers real-time network traffic analysis capabilities to identify suspicious activities, helping detect network attacks on application servers in IoT environments [4]. Research shows that DL is effective in detecting DDoS attacks, malware propagation, unauthorized access, injection attacks, and phishing attempts within network traffic, making it an important area of study [5]. However, advanced DL algorithms such as Long Short-Term Memory (LSTM) Networks, Transformers, and Encoders, which require substantial resources, face implementation challenges in low-resource IoT environments [2]. Research [4] and [5] demonstrate specially designed, innovative security solutions have the capacity to transcend traditional approaches in detecting and mitigating sophisticated network attacks in IoT infrastructures. This study seeks to address this problem by investigating the use of Shallow and Deep Feed Forward Neural Networks (FFNs) as potential solutions to enhance the security of low-resourced IoT application servers.

To address these challenges, this study examines the employability of simpler FFN models to gain insights into the protection of low-resourced IoT application servers, particularly within critical infrastructure. The subsequent sections of this paper include a literature review on DL-based IoT attack detection, followed by a methodology detailing the research

Correspondence: Niranjan Meegammana (E-mail: niranjan.meegammana@gmail.com)
Received: 16-06-2024 **Revised:** 12-08-2024 **Accepted:** 09-09-2024
Niranjan Meegammana is from Shilpa Sayura Foundation (niranjan.meegammana@gmail.com) and Harinda Fernando is from SLIIT (harinda.f@sliit.lk)
DOI: https://doi.org/10.4038/icter.v18i2.7302
The 2025 Special Issue contains the full papers of the abstracts published at the 24th ICTer International Conference.

steps. The paper then presents the results and discussion and, finally concludes with key findings and future directions.

II. Literature Review

IoT application servers enable mission-critical data aggregation, communications, and service delivery. The rapid growth of IoT in various sectors, from smart homes and healthcare to critical infrastructure, and its sheer distribution have broadened the IoT attack surface globally. This poses significant risks to data security, privacy, business, and financial stability and endangers human lives [5],[2]. Traditional security measures, such as intrusion detection systems (IDS) and firewalls, lack the capabilities to address IoT vulnerabilities, network complexities, and challenges with real-time monitoring of a large number of devices. This necessitates robust and novel approaches to security [6]. Research highlights the potential of DL in protecting networks and servers. However, their resource requirements pose crucial challenges in low-resourced IoT environments [3].

A. Deep Learning Architecture

Deep Learning (DL) models consist of interconnected nodes that allow information to flow from the input layer through hidden layers to the output layer. Neurons receiving inputs perform computations by applying an activation function to the weighted sum of inputs and generate output signals for the next layer. Through iterative processing, these models learn from data by computing loss, updating weights, and ultimately producing an output through prediction neurons [7]. Many DL models, such as Recurrent Neural Networks (RNNs), Long Short-Term Memory (LSTM) networks, and Transformers, have been effectively used in cybersecurity research [8]. However they demand substantial processing power and memory. In contrast, Feed Forward Neural Networks (FFNs) being the simplest of DL models, have been successfully employed in network attack detection in IoT environments [4]. FFNs allow the design of Shallow and Deep models by varying the depth (number of hidden layers) and breadth (number of neurons) for different tasks, as shown in Figure 2.

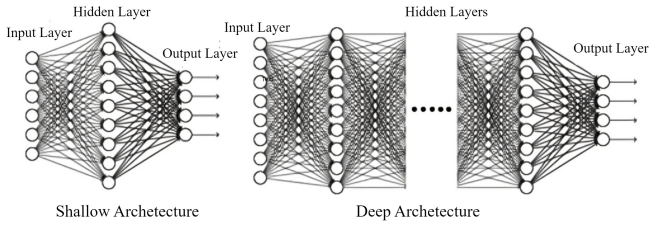

Fig. 2: Shallow and Deep Model Architectures[9].

Shallow FFN architecture reduces model complexity while learning complex patterns, using a single hidden layer with a higher number of neurons. However, it struggles with hierarchical representations in data. In contrast, Deep FFN architecture, which uses multiple hidden layers, offers enhanced performance for handling complex problems. An increased number of layers allows the model to capture intricate patterns and hierarchical representations in data. Nonetheless, their computational complexity demands higher resources [9], [10]. Hyperparameter tuning is a crucial process that involves preconfiguring important model settings before training. This significantly influences the model architecture and the learning process [11].

B. Related work

[12] systematically reviewed 69 studies focusing on DL approaches to enhance security in IoT environments. They analyze various attack types on IoT, DL architectures, and datasets, highlighting notable DL applications. They also provide a structured taxonomy highlighting research gaps and drawbacks in employing DL models for IoT security. A research gap identified in their study is the lack of discussion on model biases arising from data imbalance. [10] theoretically compare Shallow and Deep architectures, arguing that Deep models with multiple hidden layers can outperform Shallow models in classification tasks. However, the study does not consider factors such as resource usage and implementation environment, as well as validation experiments necessary to support their findings. [5] highlight the escalating security challenges in the IoT landscape, outlining common threats and the shortcomings of conventional security measures in safeguarding resource-constrained IoT devices. They underscore the potential for employing DL-based IoT security solutions, comparing them with classical machine learning approaches. However, the research falls short in providing comparative results, including implementation challenges between DL and classical machine learning methods in the IoT security context.

[13] utilized widely known KDD-Cup '99, NSL-KDD, and UNSW-NB15 datasets for network attack direction. Although the authors achieved a higher classification accuracy with the UNSW-NB15 dataset, they overlooked data balancing and hyperparameter tuning. It significantly influences the model bias and final model performance. [8] conducted a comprehensive review of 143 studies on DL approaches that address IoT security challenges. They identified various security issues in IoT requirements and examined DL approaches to address them, as well as discussed various mechanisms in studies by comparing their performances. Their review highlights the significant diversity in security requirements across studies and sectors, reflecting the concerns on the evolving landscape of IoT security that require attention. However, the study lacks a focus on the resource utilization of the proposed solutions. [14] compared the efficacy of Shallow and Deep neural network architectures to present distinct trade-offs in terms of model complexity, performance, and computational efficiency. Consequently, they state that employing them in the real world depends on specific task requirements, data complexity, and available computational resources. [15] highlight the challenges faced by IDS/IPS mechanisms in protecting IoT networks that process large volumes of information at faster speeds, as well as the heterogeneity of devices, platforms, and security

protocols.

[16] used FFNs and LSTM to create a hybrid model to detect attacks on IoT servers employing both NSL-KDD and BoT-IoT datasets achieving an accuracy of 99.95%. However, combining the BoT-IoT dataset which focuses on botnet detection and the NSL-KDD dataset which focuses on intrusion detection requires extensive preprocessing due to structural differences, complexities in addressing data normalization, and class imbalances. The methodology for merging these datasets remains unclear. [17] developed a hybrid model combining RNN and GRU algorithms, using the ToN-IoT dataset for attack classification. They achieved a high accuracy of 98% at the application layer, surpassing other DL approaches examined. However, while the RNN-GRU hybrid enhances memory and temporal capabilities, it also introduces increased model complexity and latency. This necessitates meticulous hyperparameter tuning, particularly for optimizing learning rates between RNNs and GRUs. Moreover, the computational demands of the model may pose challenges for low-resourced edge devices and could potentially lead to overfitting.

C. Research Gap

[12], [5], [8], and [15] underscore the limitations of existing IoT network attack detection methods and propose DL approaches. However, their studies often overlook crucial steps in the Machine Learning (ML) pipeline essential for model optimization, avoiding model bias, and performance improvement. [13], acknowledge the significance of the UNSW-NB15 dataset employed in this study, however, they do not effectively address the data imbalance problem in their study.

[9], [10], and [14] delve into theoretical aspects of Shallow and Deep models, but their studies lack practical experimentation in employing them for IoT network attack detection. [16] did not clarify how they merged the structurally different NSL-KDD and BoT-IoT datasets, which complicates understanding the preprocessing process and its implications for overall performance. Despite achieving high accuracy with their RNN-GRU hybrid model, [17] face challenges including overfitting, increased model complexity, latency, and high computational demands, especially for low-resourced edge devices.

This study aims to bridge these gaps by systematically building and investigating optimized Shallow and Deep DL models to safeguard IoT application servers from network attacks. The novelty of the research emerges from a structured approach to design low-resource consuming FFN models to meet limitations in IoT application servers in different environments.

III. Methodology

This study follows the Machine Learning (ML) pipeline, which is a sequence of interconnected steps designed to produce higher-quality DL models [18]. The research uses a supervised learning approach to train, validate, and test Shallow and Deep FFNs employing a labeled dataset to make predictions on new inputs.

A. Data Preparation

1) Dataset: The study utilized a subset of the widely known UNSW-NB15 dataset. The subset was created through random sampling [19]. This dataset was chosen due to its significant relevance to the network attacks on IoT servers addressed in the study. After cleaning and removing categorical variables, we balanced the label feature using random undersampling to avoid model bias toward the benign class [20]. Reducing bias is an important ethical consideration in DL tasks [21]. Ultimately, the dataset was reduced to 186,000 instances. It consisted of an equal number of attack and benign samples.

2) Feature sets: The study created two input feature sets for FFN model experimentation. The first input dataset consisted of 20 significant features identified by [4] on the same dataset. Reducing the feature set was aimed to decrease model complexity to suit low-resource IoT environments. The second input dataset consisted of 40 numerical features of the dataset, focusing on high-resource IoT environments to capture intricate relationships and hierarchical representations in data.

3) Data Normalization: Subsequently, all features except the label feature, were scaled using the min-max scaler technique, bringing features within a common range of 0 to 1. Finally, the datasets were divided into training, testing, and validation sets in a 90:5:5 ratio for model development [7]. This splitting strategy ensured allocating a substantial number of samples for training and adequate samples for validation and testing, creating a balance.

B. Model Development

The study designed two base FFN models. The Shallow model consisted of a single hidden layer with 512 neurons. It aimed to capture complex patterns and relationships within data while maintaining simplicity and computational efficiency as shown in Figure 3. The Deep model comprised 7 hidden layers employing 256, 128, 64, 32, 16, 8, and 4 neurons respectively. It allows hierarchical feature extraction to capture intricate patterns and nuances in the data as shown in Figure 3.

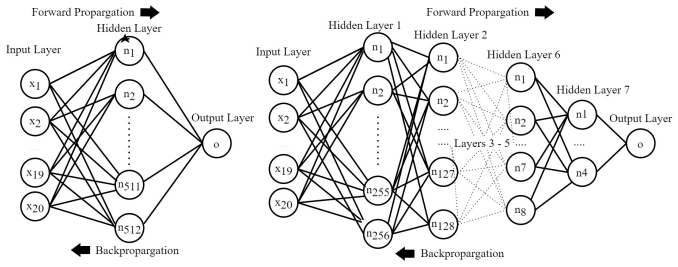

Fig. 3: Shallow and Deep base model designs

Ultimately, the study created and trained four experimental models by applying 20 and 40 feature datasets. They were named Shallow20, Deep20, Shallow40, and Deep40 to establish benchmarks across different model configurations. The models are illustrated in Figure 4.

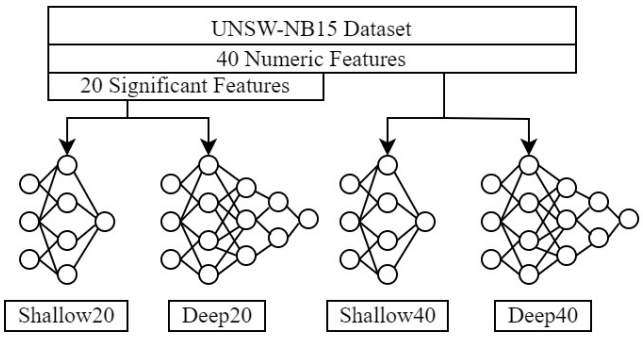

Fig. 4: Architectures of four experimental models.

The hyperparameter tuning process enhances model performance by exploring various hyperparameter combinations to select the optimal configuration for the experimental model's training. The study used Keras Tuner's random search algorithm using 50 trials. It used the training set for hyperparameters tuning while using 20% of data for internal validation to avoid overfitting [11].

Subsequently, we trained the four experimental models using their best hyperparameters, employing 20 and 40 feature training sets. The validation set was used for real-time monitoring of potential overfitting. During training, each model was saved at the epoch reaching the minimum validation loss to control overfitting, and to enhance model generalization [7].

After developing the models on Google Colab using Python, the study created four optimized executables using PyInstaller, and executed them in an isolated environment to train, build, and test each model with the relevant dataset for evaluating each model performance and resource usage [3].

C. Model Performance Evaluation

Each experimental model underwent a rigorous testing and validation process to assess its effectiveness in generalizing to new, unseen data. This unbiased evaluation was conducted to select the best-performing model using performance metrics derived from the confusion matrix, including accuracy, precision, recall, F1 score, and ROC-AUC curve. These metrics collectively offer a comprehensive evaluation of each model's ability to generalize, detect patterns, and make accurate predictions in network attack classification tasks.

D. Assesing Resource Utilizationl

The resource utilization by models during training and testing was evaluated by measuring the training and prediction times, including their CPU and memory usage. This assessment was crucial for evaluating real-time model performance and scalability in low-resourced IoT environments [3]. Furthermore, analyzing the experimental model performance metrics and resource usage helped in identifying areas for model improvement and making informed decisions about deploying them within low-resourced IoT environments to protect application servers.

IV. Results and Discussion

The four experimental models initially showed relatively high accuracies. They further improved to a range of 0.94 to 0.98 after hyperparameter tuning, raising concerns of potential overfitting. Hence, the training required close monitoring of the minimum validation loss during training.

Table 1 shows the optimized model architectures after hyperparameter tuning. In a comparison of model complexities, the Shallow20 model has the fewest (11,265) model parameters, followed by Deep20 (21,008), Shallow40 (49,313), and Deep40 (54,433) respectively.

TABLE I: Optimized architectures of models

Configuration	Shallow20	Shallow40	Deep20	Deep40
Model Parameters	11265	49313	21008	54433
Data Inputs	20	40	20	40
Hidden layers	1	1	7	7
Neurons per layer	512	512	256 to 4	256 to 4
activation_layer1	Relu	Tanh	Tanh	ReLU
activation_layer2	-	-	Tanh	ReLU
activation_layer3	-	-	ReLU	leaky_relu
activation_layer4	-	-	ReLU	ReLU
activation_layer5	-	-	ReLU	ReLU
activation_layer6	-	-	ReLU	ReLU
activation_layer7	-	-	leaky_relu	leaky_relu
optimizer	Rmsprop	Adam	Adam	Adam
learning_rate	0.001	0.001	0.001	0.001
weight_initializer	he_normal	he_normal	he_normal	glorot_uniform
batch_size	256	64	8	16
Output function	Sigmoid	Sigmoid	Sigmoid	Sigmoid

A. Training of Final Models

Validation loss is a key metric used to evaluate the performance of four models during training. It represents the error between the predicted outputs of the model and the actual outputs on the separate validation dataset used for assessing model convergence during training [11].

The study monitored the changes in training accuracy and validation loss of four models during the training process. By setting the early stopping patience value to 50 epochs, the study allowed the training to halt when the validation loss was no longer improving.

Early stopping is a regularization technique used during the training of neural network models. It helps prevent overfitting, which occurs when a model learns not only the underlying patterns in the training data but also the noise and specific details that do not generalize well to new, unseen data.

Therefore, early stopping helps mitigate this problem by halting the training process before the model starts to

overfit [11]. Figure 5 illustrates the change in validation loss of four models during training.

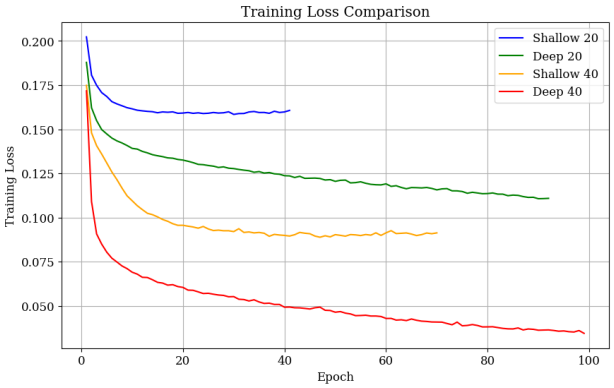

Fig. 5: Changes of Loss during Training.

Figure 6 illustrates the change in training accuracy of four models during training.

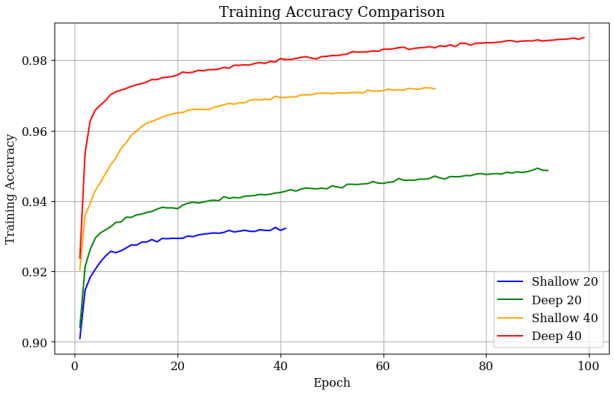

Fig. 6: Changes of Accuracy during Training.

B. Resource Utilization

The study aimed to optimize the models for performance on ARM v8 64-bit SoC processors operating at 1.5GHz with 4GB of RAM or more, assuming these represent low-resource environments. As shown in Table 4, the results demonstrated significantly lower resource utilization compared to higher-specification systems.

TABLE II: Key performance metrics of models

Performance	Shallow20	Shallow40	Deep20	Deep40
Training Time (S)	289	902	512	962
Validation Accuracy	0.93	0.94	0.97	0.98
End Epoch	10	61	39	68
Validation Loss	0.166	0.130	0.088	0.057
Prediction Time	0.55	0.56	0.57	0.59
Test Accuracy	0.93	0.94	0.97	0.98
Precision	0.94	0.96	0.97	0.98
Recall	0.92	0.92	0.97	0.98
F1 Score	0.93	0.94	0.97	0.98
ROC AUC Score	0.986	0.990	0.996	0.998
Average Precision	0.987	0.991	0.996	0.998

C. Model Comparison

As shown in Table 4, the four experimental models achieved relatively high accuracies between 0.93 and 0.98, alongside high precision, recall, F1 score, and ROC AUC scores, indicating their strong predictive capabilities. Among them, the Deep40 model demonstrated superior performance across multiple metrics, achieving the highest test accuracy of 0.98, validation accuracy of 0.98, precision of 0.981, F1 score of 0.98, ROC AUC score of 0.998, and average precision of 0.998, while recording the minimum validation loss of 0.057.

Hence, the Deep40 model consistently outperformed the other models, indicating its robustness in making accurate predictions and its exceptional ability to generalize to unseen data. Exhibiting the highest number of true positives (4572) and true negatives (4537), the Deep40 model further demonstrates its effectiveness in correctly classifying both positive and negative instances. Moreover, exhibiting the lowest number of false positives (78) and false negatives (113) showcases its accuracy in making predictions with high sensitivity and specificity.

D. Resource Utilization

The Shallow20 model with simpler architecture and lower computational complexity, achieved faster training time. This factor is insignificant if the model training time is not a critical factor for the task. The difference in prediction times between the Shallow20 (0.55 seconds) and the Deep40 (0.59 seconds) model may appear small. However, even minor differences can be significant in real-time circumstances where predictions are made continuously or in batches in low-resourced IoT environments.

TABLE III: Key performance metrics of models

Performance Data	Shallow20	Shallow40	Deep20	Deep40
Model Size (KB)	110	190	648	708
Prediction Time (s)	0.55	0.56	0.57	0.59
CPU Usage (%)	0.25	0.26	0.30	0.32
Memory Usage (GB)	0.016	0.017	0.019	0.021

As shown in Table 4, the single-layer Shallow20 model exhibited the lowest CPU and memory usage compared to deep models with multiple layers and more parameters. The Shallow20 model requires less RAM for storing and processing model parameters and intermediate computations, contributing to their lower memory consumption. During model inference, the Shallow20 model offers advantages for deployment in resource-constrained environments and time-sensitive real-time IoT applications, where timely responses are necessary to mitigate security threats effectively.

The Deep40 model exhibits superior performance and better overall model quality, particularly effective for classification tasks in well-resourced IoT servers. By recognizing these differences and trade-offs, researchers and practitioners can make informed decisions regarding model selection and deployment strategies for enhancing security mechanisms for IoT application servers.

V. Conclusion

This research focused on enhancing IoT application server security using Shallow and Deep FFN models employing two datasets followed by a systematic approach involving iterative improvements. The investigation revealed that despite the higher computational demands, the Deep40 FFN model using a 40 features dataset demonstrated superior accuracy and discriminative capabilities for detecting network attacks on IoT application servers.

Conversely, the Shallow20 model which uses 20 significant features, performed with lower resource requirements with slightly lower accuracy. Hence, it remained suitable for low-resource IoT environments demanding low-latency trading-off accuracy. Furthermore, the analysis showed the significance of feature selection, data balancing, and hyperparameter tuning for optimizing model performance. In conclusion, the study underscores the practical implications of employing varied FFN architectures to enhance low-resourced IoT applications server security.

Finally, the choice between Shallow and Deep FFN models depends on specific application requirements, considering trade-offs in computational resources and model complexity, inference speed, interpretability, scalability, and cost. This research significantly advances low-resourced IoT application security by demonstrating the effectiveness of Shallow and Deep FFN architectures. It proposes a novel and practical deployment strategy by integrating the Deep40 model into next-generation firewall systems to protect higher-value IoT servers and suggesting future directions for exploring Shallow-Deep hybrid FFN models in defending edge servers, in low-resourced IoT environments.

References

[1] A. Khang, V. Abdullayev, V. Hahanov, and V. Shah, *Advanced IoT Technologies and Applications in the Industry 4.0 Digital Economy*. CRC Press, 2024.

[2] A. Alwarafy, K. A. Al-Thelaya, M. Abdallah, J. Schneider, and M. Hamdi, "A survey on security and privacy issues in edge computing-assisted internet of things," *IEEE Internet of Things Journal*, pp. 1–1, 2020.

[3] N. Moustafa, M. Abdel-Basset, and R. Mohamed, *Deep Learning Approaches for Security Threats in IoT Environments*. Wiley-IEEE Press, 2022.

[4] N. Meegammana and H. Fernando, "Detection of network attacks on application servers using deep learning in iot environments," *None*, 2023.

[5] A. Sagu, N. S. Gill, and P. Gulia, "Artificial neural network for the internet of things security," *International Journal of Engineering Trends and Technology*, vol. 68, no. 11, pp. 129–136, 2020.

[6] E. Schiller, A. Aidoo, J. Fuhrer, J. Stahl, M. Ziörjen, and B. Stiller, "Landscape of iot security," *Computer Science Review*, vol. 44, no. 44, p. 100467, 2022.

[7] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[8] Y. Ali, H. U. Khan, and M. Khalid, "Engineering the advances of the artificial neural networks (anns) for the security requirements of internet of things: A systematic review," *Journal of Big Data*, vol. 10, no. 1, 2023.

[9] I. H. Sarker, "Ai-based modeling: Techniques, applications and research issues towards automation, intelligent and smart systems," *SN Computer Science*, vol. 3, no. 2, 2022.

[10] M. Bianchini and F. Scarselli, "On the complexity of neural network classifiers: A comparison between shallow and deep architectures," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 8, pp. 1553–1565, 2014.

[11] F. Chollet, *Deep Learning with Python*. Manning, 2018.

[12] L. Aversano, M. L. Bernardi, M. Cimitile, and R. Pecori, "A systematic review on deep learning approaches for iot security," *Computer Science Review*, vol. 40, p. 100389, 2021.

[13] S. Choudhary and N. Kesswani, "Analysis of kdd-cup'99, nsl-kdd and unsw-nb15 datasets using deep learning in iot," *Procedia Computer Science*, vol. 167, pp. 1561–1573, 2020.

[14] C. C. Aggarwal, *Neural Networks and Deep Learning*. Springer Nature, 2023.

[15] J. Cook, S. Rehman, and M. Khan, "Security and privacy for low power iot devices on 5g and beyond networks: Challenges and future directions," *IEEE Access*, vol. 11, pp. 39 295–39 317, 2023.

[16] O. Jullian, B. Otero, E. Rodriguez, N. Gutierrez, H. Antona, and R. Canal, "Deep-learning based detection for cyber-attacks in iot networks: A distributed attack detection framework," *Journal of Network and Systems Management*, vol. 31, no. 2, 2023.

[17] M. Khan, N. Khan, M. Alshehri, A. Alazeb, S. Ullah, N. Naz, and J. Ahmad, "A hybrid deep learning-based intrusion detection system for iot networks," *Mathematical Biosciences and Engineering*, vol. 20, no. 8, pp. 13 491–13 520, 2023.

[18] H. Hapke and C. Nelson, *Building Machine Learning Pipelines : Automating Model Life Cycles with Tensorflow*. O'Reilly, 2020.

[19] W. David, "UNSW-NB 15 Dataset," https://www.kaggle.com/datasets/mrwellsdavid/unsw-nb15, 2019.

[20] F. Thabtah, S. Hammoud, F. Kamalov, and A. Gonsalves, "Data imbalance in classification: Experimental evaluation," *Information Sciences*, vol. 513, pp. 429–441, 2020.

[21] N. Sutaria, "Bias and ethical concerns in machine learning," https://www.isaca.org/resources/isaca-journal/issues/2022/volume-4/bias-and-ethical-concerns-in-machine-learning, 2022.