Eur. Phys. J. C manuscript No.
(will be inserted by the editor)

Enhancing Attack Detection on Edge Servers with Hybrid Neural Networks

Edge Security with Shallow-Deep Hybrid Strategies

Niranjan W. Meegammana [a,b], **Harinda Fernando**[c,d]

[1]Niranjan W Meegammana Shilpa Sayura Foundation, Kandy, Sri Lanka
Tel.: +94-777573857
Fax: +94-812575759
E-mail: niranjan.meegammana@gmail.com
[2]Harinda Fernando Sri Lanka Institute of Information Technology, Malabe, Sri Lanka
Received: date / Accepted: date

Abstract Neural Networks (NNs) excel at learning complex patterns but are often resource-intensive and prone to overfitting. In contrast, shallow models are more efficient but struggle to handle complex data. Hybrid models can leverage the strengths of both approaches, providing superior performance, adaptability, and resource efficiency in network attack detection. This study investigates Shallow-Deep Hybrid Fusion NN models specifically for detecting network attacks on edge servers. The research evaluates 12 hybrid fusion models that combine a single-layer shallow model with 512 neurons and a seven-layer deep model with varying neuron counts, utilizing datasets with 20 and 40 features. The performance results and resource usage metrics reveal that the hybrid models outperform their individual shallow and deep counterparts. Among the fusion models, the 20-feature maximum and minimum hybrid models achieved 95.00% accuracy and a precision score of 0.97, demonstrating their suitability for resource-constrained edge servers. In contrast, the 40-feature concatenation and maximum hybrid models attained 98.0% accuracy and a precision score of 0.99, indicating their appropriateness for high-resource edge server environments. Future research will explore integrating these models with federated learning for autonomous vehicles, which may extend their applications beyond cyber security.

Keywords Shallow-Deep · Parallel Fusion · Hybrid Neural Networks · Attack Detection · Edge Servers

[a]e-mail: niranjan.meegammana@gmail.com
[b]Shilpa Sayura Foundation, Kandy, Sri Lanka
[c]e-mail: harinda.f@sliit.lk
[d]Sri Lanka Institute of Information Technology, Malabe, Sri Lanka

1 Introduction

The Internet of Things (IoT) consists of interconnected devices that exchange data over the internet. They range from smart home appliances to industrial sensors, that enable automation, real-time data analysis, and enhanced user experiences. However, the large number of devices and volume of data they generate poses significant challenges that require edge computing.

This edge network architecture facilitates distributed computing by positioning resources closer to data sources. It helps in reducing latency for applications that need near-real-time responses. It helps avoid network congestion and reduces costs by processing data locally and sending only relevant information to the cloud. Additionally, keeping sensitive data closer to the source enhances security and privacy while improving system reliability by allowing devices to function independently of a stable internet connection. Despite its advantages, edge computing faces unique challenges. Edge devices often have limited processing power, memory, and storage compared to cloud servers, necessitating lightweight algorithms. The diversity of IoT ecosystems complicates the creation of uniform solutions, while edge devices are more vulnerable to physical attacks and data breaches, requiring robust security measures. Furthermore, managing a large number of distributed devices poses challenges, especially regarding energy efficiency in resource-constrained environments. [1].

This research investigates the effectiveness of a Shallow-Deep Hybrid Fusion Neural Network solution in addressing network attacks on edge servers. Figure 1 illustrates the layered architecture of an edge environment.

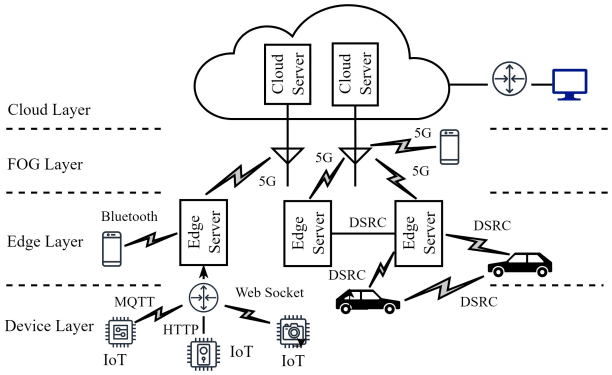

Fig. 1 Architecture of an edge environment

1.1 Edge environments security

Edge computing, evolving with the Internet of Things (IoT), decentralizes crucial computational resources such as application servers, IoT devices, sensors, mobile devices, and users. Edge servers are closely integrated with IoT devices, processing data at or near the device level. This enables seamless data collection, aggregation, processing, and communication with cloud servers. This architecture enhances functional latency, reduces network bandwidth, improves privacy, and increases survivability [2]. Gartner projects that by 2025, over 75 percent of enterprise data will be generated and processed outside the cloud due to the rapid expansion of edge infrastructures [3].

The rapid proliferation of edge networks introduces significant security challenges for edge applications and data. Attackers may exploit vulnerabilities in IoT devices within these networks to carry out various attacks. For instance, the 2016 Mirai botnet leveraged unsecured IoT devices to execute a massive DDoS attack, affecting major services like DNS provider Dyn and numerous websites globally. Unsecured IoT devices on edge networks can initiate large-scale DDoS attacks, overwhelming edge servers and rendering them unavailable to legitimate users. Notable examples include the 2016 DDoS attacks on popular services like Spotify, Twitter, and Netflix, which were traced back to compromised IoT devices [4]. Lateral attacks can also move across an edge network from one device to another, exploiting weak security measures to access critical edge servers. A prominent case is the 2019 attack on the City of New Orleans, where attackers disrupted operations and demanded a ransom. In addition to DDoS attacks, other threats such as data exfiltration and injection attacks pose serious risks. Data exfiltration involves stealing sensitive information for purposes like espionage or identity theft. For example, the 2017 Equifax breach compromised the personal data of millions, potentially leading to identity theft. Injection attacks, such as the 2014 SQL injection on CardSystems Solutions, exploited software vulnerabilities to steal data [4]. The rise of 5G networks amplifies these security concerns due to increased speed, connectivity, and attack surfaces. Protecting edge servers is particularly challenging because of their limited processing power, extensive distribution, and the lack of universal security standards. Traditional security mechanisms struggle with the unique and open architecture of edge networks that lack perimeter protection. Furthermore, the heterogeneity of devices complicates the standardization of security protocols [5] [6]. Addressing these risks requires novel approaches to safeguard applications, devices, and data in edge environments.

Security breaches on edge networks can lead to significant remediation costs, legal fees, regulatory fines, revenue loss, and damage to brand reputation. High-profile breaches can erode stakeholder trust and undermine confidence in organizational security practices. Organizations bound by regulations like GDPR and HIPAA risk legal consequences and penalties for data breaches. Geopolitical tensions, such as the Russia-Ukraine conflict, may encourage attacks on critical infrastructure, potentially disrupting utility services and escalating into cyber warfare [10]. These severe risks highlight the need for robust security measures to protect edge servers, ensuring the integrity and continuity of services and safeguarding societal functioning and economies [11].

Traditional security measures, such as firewalls, antivirus solutions, and Intrusion Detection Systems (IDS), face significant challenges in edge computing and IoT environments. Edge devices with limited processing power, memory, and energy, making it difficult to run robust security software without degrading performance. Moreover, the IoT ecosystem comprises a diverse array of devices from various manufacturers, complicating the application of standardized security measures and necessitating device-specific protocols.

The lack of universally accepted security standards further hampers their interoperability. It is challenging to enforce consistent security policies across the edge network. Due to this fragmentation introduces vulnerabilities, as devices may not receive timely updates or patches. The variability in device capabilities also complicates compliance monitoring, increasing the risk of successful attacks.

With evolving cyber threats, traditional security solutions that rely on known threat signatures that can be ineffective against sophisticated attacks. Edge devices, deployed in diverse environments, are vulnerable to physical tampering and unauthorized access. Hence perimeter-based security is less effective. Addressing

these challenges, require adopting robust security approaches tailored to edge devices' specific needs. This includes implementing lightweight security solutions and leveraging advanced technologies, such as deep learning, that can further enhance threat detection and response capabilities while accommodating resource constraints[]

1.1.1 Edge environments security challenges

Edge computing, evolving with the Internet of Things (IoT), decentralizes crucial computational resources such as application servers, IoT devices, sensors, mobiles, and users [1] . Local edge servers connect and manage IoT devices, enabling seamless data collection, aggregation, processing, and communication with cloud servers. This architecture enhances functional latency, reduces network bandwidth, improves privacy, and survivability [2]. Gartner projects that by 2025, over 75 percent of enterprise data will be generated and processed outside the cloud due to the rapid expansion of edge infrastructures [3].

The rapid proliferation of edge networks introduces significant security challenges for edge applications and data. Attackers may exploit vulnerabilities in IoT devices in these networks to carry out various attacks. For example, the 2016 Mirai botnet leveraged unsecured IoT devices to execute a massive DDoS attack, affecting major services like DNS provider Dyn and numerous websites globally. Lateral attacks can move across an edge network from one device to another, exploiting weak security measures to access critical edge servers, as seen in the 2019 attack on the City of New Orleans, where attackers disrupted operations and demanded a ransom [4]. Other threats include DDoS attacks, data exfiltration, and injection attacks, such as identity theft.

Unsecured IoT devices on edge networks can initiate large-scale DDoS attacks, overwhelming edge servers and making them unavailable to legitimate users. Data exfiltration involves stealing sensitive information, for purposes such as espionage or identity theft. For instance, the 2017 Equifax breach, compromised the personal data of millions, potentially leading to identity theft [4]. Injection attacks, like the 2014 SQL injection attack on CardSystems Solutions, exploited software vulnerabilities to steal data [4]. The rise of 5G networks amplifies security concerns due to increased speed, connectivity, and attack surface [5]. Protecting edge servers is challenging due to limited processing power, extensive distribution, and a lack of universal security standards. Traditional security mechanisms struggle with the unique and open architecture of edge networks that lack perimeter protection. Moreover, the heterogeneity of devices also complicates the standardization of security protocols [5] [6]. Addressing these risks requires novel approaches to safeguard applications, devices, and data in edge environments.

Attacks on edge networks can disrupt critical services, compromise vital information, cause financial losses, and pose serious risks to operations [7]. Notable examples include the 2016 DDoS attacks on Spotify, Twitter, and Netflix traced back to compromised IoT devices [8]. As edge servers handle sensitive information, they are prime targets for cyber attacks through connected IoT devices. These breaches disrupt operations result in significant financial losses and endanger public safety. In sectors like healthcare and automotive, compromised edge devices can lead to life-threatening situations, and attacks on environmental monitoring servers can have devastating consequences[9].

Security breaches on edge networks can lead to significant remediation costs, legal fees, regulatory fines, revenue loss, and damage to brand reputation. High-profile breaches can erode stakeholder trust and undermine confidence in organizational security practices. Organizations bound by regulations like GDPR and HIPAA risk legal consequences and penalties for data breaches. Geopolitical tensions, such as the Russia-Ukraine conflict, may encourage attacks on critical infrastructure, potentially disrupting utility services and escalating into cyber warfare [10]. These severe risks highlight the need for robust security measures to protect edge servers, ensuring the integrity and continuity of services and safeguarding societal functioning and economies [11].

Traditional security measures, such as firewalls, antivirus solutions, and Intrusion Detection Systems (IDS), face significant challenges in edge computing and IoT environments. Edge devices with limited processing power, memory, and energy, making it difficult to run robust security software without degrading performance. Moreover, the IoT ecosystem comprises a diverse array of devices from various manufacturers, complicating the application of standardized security measures and necessitating device-specific protocols.

The lack of universally accepted security standards further hampers their interoperability. It is challenging to enforce consistent security policies across the edge network. Due to this fragmentation introduces vulnerabilities, as devices may not receive timely updates or patches. The variability in device capabilities also complicates compliance monitoring, increasing the risk of successful attacks.

With evolving cyber threats, traditional security solutions that rely on known threat signatures that can be ineffective against sophisticated attacks. Edge de-

vices, deployed in diverse environments, are vulnerable to physical tampering and unauthorized access. Hence perimeter-based security is less effective. Addressing these challenges, require adopting robust security approaches tailored to edge devices' specific needs. This includes implementing lightweight security solutions and leveraging advanced technologies, such as deep learning, that can further enhance threat detection and response capabilities while accommodating resource constraints [7].

1.1.2 Neural networks for edge security

Research indicates that Neural Networks (NNs) effectively detect cyber threats in real time, extending conventional machine learning with high precision in attack classification and malware pattern recognition [12]. NN models with their layered architecture, provide flexibility to capture patterns in network traffic data. They address issues, like anomaly detection, intrusion detection, threat intelligence, behavioral analysis, privacy preservation, and adaptive defense in edge environments. For example, [7] demonstrates that Feed Forward Neural Networks (FFNs) achieve high accuracy and precision in detecting network attacks on IoT application servers.

Hybrid Neural Networks (NNs) combine multiple models to enhance problem-solving capabilities by leveraging each model's strengths to address others' weaknesses. This approach improves predictive performance, reduces overfitting, and increases robustness and resilience. Hybrid models also offer greater flexibility in designing NN architectures, enhancing interpretability. Traditional techniques like bagging, stacking, and boosting are commonly used, while newer methods such as weighted average, concatenation, and multiplication create more versatile hybrid architectures [13].

Numerous studies have examined hybrid NN models in various domains [14] [15]. However, research on Shallow-Deep Hybrid Fusion approaches in cyber security is limited. This study aims to fill this gap by exploring how Shallow-Deep Hybrid Fusion models can enhance edge server security against evolving cyber threats. The research seeks to advance knowledge to strengthen edge server security by using hybrid NNs and evaluating their effectiveness in combating network attacks.

This paper is organized into five sections. It begins with an introduction to the research context, followed by a literature review on security risks in edge environments, NN-based attack detection, and existing gaps. The methodology section provides details of the design, dataset, and approach, including the development and evaluation of 12 Shallow-Deep ANN Hybrid Fusion models. The findings section presents the experimental results and model performance, offers insights, and outlines future research directions. The paper concludes with a summary of the research.

2 Literature review

2.1 Neural network solutions

Neural Networks (NNs) have proven effective in detecting network attacks on application servers [7]. Their ability to analyze network traffic patterns and anomalies enables real-time identification of suspicious activities. NNs can enhance the security of edge applications and adapt by learning from new attacks and data, supporting proactive defense against emerging threats [18].

NNs face several challenges despite their advantages for network security. They require large amounts of high-quality, labeled data, which is costly and time-consuming. Training NNs demands significant computational resources and time. They are also prone to overfitting and underfitting, affecting their ability to generalize to new data. Moreover, NNs often function as black boxes, making it difficult to interpret their decisions and outcomes [17].

[19] highlight that NN models are vulnerable to adversarial attacks like data poisoning, evasion attacks, and model inversion, leading to incorrect predictions. Therefore, it requires developing robust models. Additionally, the lack of transparency in NNs raises ethical concerns about privacy, bias, and fairness underscoring the need for representative training data, including accountability and transparency [20].

2.2 Similar work

Several studies explore the critical balance between performance and security in edge computing. For instance, [22] compares edge computing with traditional cloud computing, focusing on network architectures and optimal server placement in cloud-edge environments. This study underscores the advantages of edge computing in latency-sensitive applications like IoT and 5G. However it does not explore the challenges of heterogeneous device management and security risks such as distributed monitoring and data privacy in edge networks. Similarly, [23] explores the integration of edge computing with 5G, especially in applications like autonomous driving and augmented reality (AR). They do not discuss the significance of secure communication and privacy protection in these environments. Both of these studies emphasize the benefits of lower latency and bandwidth efficiency. However, they do not provide a clear

roadmap for addressing the inherent security challenges, particularly in real-time edge deployment scenarios. [8] and [24] take their discussions a step further by focusing on specific security concerns in edge environments. [8] examines privacy issues and attack vectors in edge-assisted IoT, proposing potential solutions but lacking concrete case studies or real-world applications to demonstrate the feasibility of their approaches. [24] approach is more focused discussing the risks of malware targeting industrial edge servers. They propose a CNN-based real-time malware detection system. This research highlights the physical risks involved, particularly for industrial systems, but it is limited by issues such as class imbalance in the datasets used, which can result in bias in attack detection.

With regards to using hybrid neural network (NN) models to address the complex nature of edge network threats, [28], [29], and [35] explore different approaches. [28] provides a thorough exploration of hybrid deep learning models, particularly focusing on ensembling techniques like bagging, boosting, and decision fusion. These approaches combine multiple models to leverage their strengths, improving the overall performance of IDSs. However, [35] highlights that while hybrid models offer performance benefits, they face challenges in terms of interpretability and adaptability, especially in dynamic edge environments. The study also underscores adversarial security concerns but lacks concrete metrics to evaluate the effectiveness of the proposed hybrid approaches. On a more technical level, [35] discusses the complexity of NN model fusion in edge environments, detailing methods such as feature-level and decision-level fusion. These methods involve combining predictions from different models to improve accuracy and reliability, but as the study notes, they introduce significant computational complexity, making them difficult to implement in resource-constrained edge environments. While these studies present promising approaches to hybrid model development, they collectively highlight the need for practical evaluation metrics, such as inference time, power consumption, and computational efficiency, especially for real-time applications.

Deep learning (DL) techniques have become essential in developing robust intrusion detection systems for edge networks. Several studies, such as [26] , [27], and [34], investigate various DL models like CNNs, RNNs, and BiLSTMs for detecting a wide range of attacks. [26] offers a broad review of DL techniques for network and host-based intrusion detection, covering architectures like RNN, CNN, DBN, GAN, and Autoencoders. The study provides valuable insights into the advantages of each architecture but lacks a discussion on practical implementations or solutions that can be applied in real-world scenarios. In contrast, [27] focuses on BiLSTM models for detecting DDoS attacks, explaining that while BiLSTMs can capture bidirectional patterns in data, they also introduce complexity and longer training times. The study could benefit from including additional metrics such as CPU/memory usage, F1-scores, and training times to provide a more well-rounded comparison with simpler models like RNNs or LSTMs. Additionally, [34] takes a hybrid approach by combining RNNs and GRUs for attack classification, achieving high accuracy rates in both the application and network layers when tested with the ToN-IoT dataset. However, this increased complexity comes with trade-offs, particularly in terms of computational demands, which may prove challenging for resource-limited edge devices. These studies collectively demonstrate that while DL models show promise in intrusion detection for edge networks, the balance between model complexity and computational efficiency remains a key challenge, especially in environments where real-time threat detection is critical.

Hybrid models, which combine multiple neural network architectures, have shown significant potential in enhancing security for edge computing environments. Studies such as [32], [33], and [36] explore various hybrid models and their applications. [32] investigates the use of CNN-LSTM hybrids to create Shallow-Deep models, using techniques like concatenation, weighted averages, and maximum-minimum fusion to improve accuracy. The research demonstrates that hybrid models can achieve superior performance in detecting attacks; however, it also introduces additional computational overhead, which is a concern in latency-sensitive environments. Similarly, [33] explores hybrid models such as Cu-GRU LSTM and Cu-DNN LSTM, specifically focusing on their application in low-resourced IoT devices. While these models offer performance gains, the study emphasizes the importance of evaluating metrics like power consumption, CPU/memory usage, and latency to ensure their practicality in real-world deployments. Without such evaluations, hybrid models may introduce complexity that outweighs their benefits in constrained environments. [36] highlights a similar issue, proposing a federated learning-based hybrid approach to reduce false detection rates and defend edge networks from zero-day attacks. Their novel use of a trusted security game approach, where predictions are made cooperatively across edge servers, offers a fresh perspective on distributed security. However, like many federated learning systems, this approach is vulnerable to model poisoning from malicious nodes, a risk that the study does not fully address. These studies highlight the ongoing exploration of hybrid models for edge se-

curity, emphasizing their potential but also pointing to the critical need for a thorough evaluation of their real-world applicability, particularly in terms of resource usage and vulnerability to adversarial attacks.

2.3 Research gaps

Several studies have explored neural networks (NNs) for network attack detection. However, gaps still need to be addressed in their applicability to resource-constrained environments like edge servers. While [2] and [22] discuss edge computing architecture and deployment challenges, they overlook security implications such as data balancing and hyperparameter tuning. [7] highlights the need for NN based solutions for addressing emerging security challenges, while [17] discusses NN challenges in network attack detection. [19] raises concerns about adversarial attacks on NN models, while [8] discusses increased risks to data security and privacy in edge computing. [24] proposes a CNN-based solution for malware detection but overlooks dataset class imbalances and data merging clarity. [26] reviews deep learning for intrusion detection but lacks practical deployment insights, and [27] explores hybrid techniques without practical comparisons. [33] highlights the benefits of hybrid models but lacks a model selection strategy. [34] achieves high accuracy with RNN-GRU hybrids but may face computational challenges in resource-limited environments. [29] discusses hybrid models, but lacks practical applications for edge networks, while [30] implements federated learning but does not address scalability or security concerns. [33] shows strong results with LSTM-GRU hybrids but lacks detailed resource utilization metrics. [36] proposes federated hybrid models but misses risks like malicious nodes and model poisoning. [25] implements a distributed deep learning model using FFNs and LSTMs, achieving high accuracy but failing to address deployment complexities in heterogeneous edge environments with varying hardware capabilities. LSTMs, known for their computational intensity, may be impractical in latency-sensitive applications like autonomous vehicles.

[36] proposes federated hybrid models but misses risks like malicious nodes and model poisoning. [25] implements a distributed deep learning model using FFNs and LSTMs, achieving high accuracy but failing to address deployment complexities in heterogeneous edge environments with varying hardware capabilities. LSTMs, known for their computational intensity, may be impractical in latency-sensitive applications like autonomous vehicles.

Hybrid models that combine shallow and deep architectures offer a balanced approach, addressing the computational limitations of edge servers while maintaining accuracy. However, prior studies such as [26] and [29] primarily focus on ensemble methods like bagging and boosting, neglecting key edge computing challenges. These studies often overlook parallel model fusion techniques, such as weighted averaging and concatenation, which are critical for optimizing resource efficiency in edge environments. Consequently, more comprehensive studies are required to address these gaps, particularly for network attack detection on edge application servers using Shallow-Deep hybrid models.

This research specifically addresses these gaps by focusing on Shallow-Deep Hybrid Fusion models for edge security. It investigates various parallel fusion techniques, including weighted averaging, concatenation, maximum, minimum, multiplication, and subtraction, while prioritizing low memory and CPU usage. This approach ensures that the models remain suitable for real-world edge deployment. In addition to accuracy, precision, recall, and F1 score, this research also evaluates resource utilization performance, providing a well-rounded solution for attack detection in edge environments.

3 Methodology

The study employed an ML pipeline, as depicted in Figure 2, consisting of interconnected steps to produce a high-quality final product [37]. This approach builds on [15] by developing Shallow-Deep Hybrid Fusion models to improve security on edge application servers against network attacks.

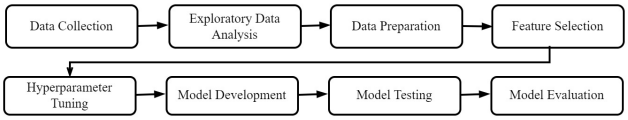

Fig. 2 Research process

3.1 Data collection

The research assessed the CICIDS2017, Bot-IoT, and UNSW-NB15 datasets and selected UNSW-NB15 for its comprehensive and diverse coverage, making it suitable for a broad spectrum of network intrusion detection tasks [15]. This study used a subset of the UNSW-NB15 dataset, containing 257,673 network traffic instances with 45 features across 9 attack classes and normal data. It employed a supervised learning approach using the target feature "label," categorized as "attack" or "benign" classes [38]. TTo address computational intensity, and reduce training time, a subset of data was chosen and balanced the classes through random undersampling [39]. It finally reduced the dataset to 186,000

instances. The UNSW-NB15 dataset consists of 9 attack categories, comprising 50% of the samples, with the remaining 50% being benign instances, as described in Table I.

The UNSW-NB15 dataset consists of 9 attack categories and begin instances described in Table 1.

Table 1 Attack categories in the dataset

attack	description
Fuzzers	Pose a threat to IoT applications by injecting random data, potentially leading to processing failures or causing application servers to crash.
Analysis	Conducts active reconnaissance, port scans, and vulnerability assessments to identify potential vulnerabilities for exploitation.
Backdoor	Install malicious software to establish unauthorized remote access to application servers.
Denial of Service (DoS)	Exploit cryptographic weaknesses to target poorly encrypted data blocks, streams, or messages.
Generic	Leverage cryptographic principles to target weakly encrypted data blocks, streams, or messages.
Reconnaissance	Passively gather preliminary information about target hosts and utilize publicly accessible data sources for intelligence gathering.
Shellcode	Inject payloads to compromise target systems, granting attackers remote access to a command shell within application servers.
Worms	Spread malware via networks, converting infected IoT devices into botnet zombies for executing distributed attacks.
Exploits	Exploit known vulnerabilities in operating systems or applications to gain unauthorized access for malicious activities.

The dataset includes traffic samples representing diverse network threats such as unauthorized access, data interception, privilege escalation, data theft, and system disruption, posing significant risks to edge environment application servers as shown in Table I.

Figure 3 illustrates the distribution of 9 attack types described in table 1.

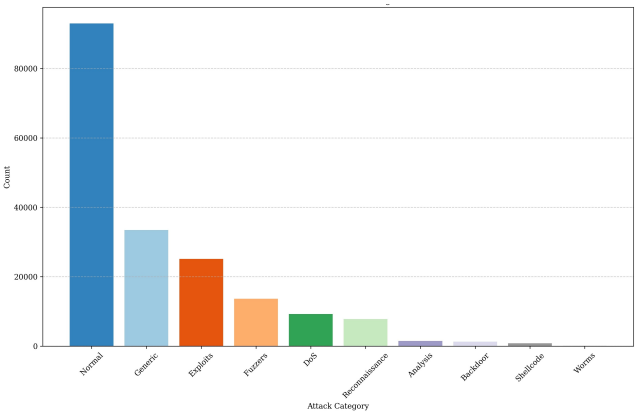

Fig. 3 Distribution of attacks and label classes

Figure 4 illustrates the distribution of labels in the target variable after class balancing

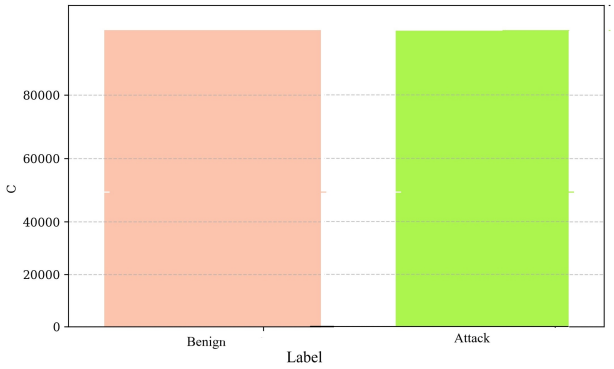

Fig. 4 Distribution of label classes

3.2 Data splitting

The study allocated 90% of data to training to improve model performance by providing more examples. Assigning 5% of the data for testing offered an independent evaluation of the model's generalization. Another 5% was used for validation to monitor performance and prevent overfitting during training. This data-splitting strategy ensured adequate training data, robust evaluation, and effective performance monitoring [39].

3.3 Feature selection

The study used two datasets derived from the preprocessed UNSW-NB15 dataset. The first set consisted of 20 significant features selected through amalgamation of multiple techniques by [7], while the second set included 40 numeric features of the dataset. Feature selection aims to investigate the effects of low and high-dimensional data on model performance in different environments. Due to the reduced model complexity, a dataset featuring 20 features may be suitable for attack detection on low-resource edge servers. Likewise,

the dataset comprising 40 features might be deployed in high-end edge servers. Table 2 lists these feature sets.

Table 2 Feature sets used in the study

feature set	features
20 significant features	dload, sbytes, sttl, smean, rate, ct_srv_dst, sload, ct_dst_src_ltm, dbytes, ackdat, dttl, dinpkt, ct_srv_src, tcprtt, dur, synack, ct_dst_sport_ltm, dmean, and sinpkt [38].
40 numeric features	id, dur, rate, sbytes, dbytes, dpkts, spots, sload, dload, sloss, dloss, sinpkt, dinpkt, sjit, djit, swin, dwin, sttl, dttl, smean, dmean, tcprtt, synack, ackdat, trans_depth, response_body_len, ct_srv_src, ct_state_ttl, ct_dst_ltm, ct_src_dport_ltm, ct_dst_sport_ltm, ct_dst_src_ltm, ct_src_ltm, ct_srv_dst, is_ftp_login, ct_ftp_cmd, ct_flw_http_mthd, and is_sm_ips_ports [38].

3.4 Data normalization

The study used Min-max normalization for feature scaling to preserve data distribution, handle bounded features, improve model convergence, and prevent feature dominance. Min-max scaling normalization transforms feature values to a common scale, typically between 0 and 1, ensuring equal contribution during model training and evaluation [17]. Figure 5 illustrates the effect of normalization on dataset features.

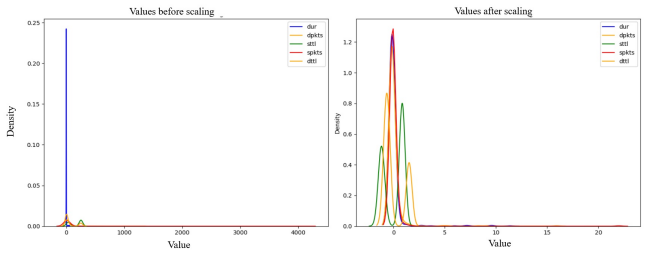

Fig. 5 Data values before and after normalization

Min-Max Scaling applies the following formula to each feature in the dataset:

$$x_{\text{scaled}} = \frac{x - \min(x)}{\max(x) - \min(x)} \quad (1)$$

where:

- x is the original value of the feature.
- $\min(x)$ is the minimum value of the feature x across the dataset.
- $\max(x)$ is the maximum value of the feature x across the dataset.
- x_{scaled} is the scaled value of the feature x.

3.5 Shallow and deep model architectures

The study used shallow and deep neural network (NN) models to develop hybrid fusion models [35] Model Architectures and Hyperparameter Space. The shallow NN model uses a single hidden layer with 512 neurons, while the deep NN has a more complex architecture with 7 hidden layers consisting of 256, 128, 64, 32, 16, 8, and 4 neurons. Using 20 and 40 features, four base NN models were designed as shown in Table 2.

All four models Shallow20, Shallow40, Deep20, and Deep40 shared the same hyperparameter space for the output layer, activation functions, learning rate, weight initializer, optimizer, loss function and batch size. During hybrid model training, the study used the Adam optimization algorithm to adjust weights and minimize errors by updating the network based on gradients of the loss function. The study also used the Binary Cross-Entropy Loss (Log Loss) function to measure how well predictions match actual labels and guide optimization to improve model performance [39]. The study employed Keras Random Search, focusing on these parameters as shown in Table 2 to find the optimal hyperparameters for each respective model.Table 3 illustrates the architecture and hyperparameter space of the four Shallow and Deep models.

Table 3 Model architectures and hyperparameter Space

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	256 128 64 32 16 8 4	256 128 64 32 16 8 4
Output Layer	Sigmoid function			
Activation functions	ReLU, Tanh, and Leaky ReLU			
Weight initializer	RMSprop, he_normal, glorot_uniform			
Optimizer	Adam			
Loss function	Binary Cross-Entropy Loss (Log Loss)			
Batch size	8, 16, 32, 64, 128, 256			
Learning rate	0.1, 0.01, 0.001			

3.6 Hybrid fusion model creation workflow

The study developed four base models: Shallow20, Deep20, Shallow40, and Deep40, utilizing 20 and 40 feature sets , then created optimized shallow and deep models with their best hyperparameters. Subsequently by combining the Shallow20-Deep20 and Shallow40-Deep40 models using decision fusion functions, including weighted

averaging, concatenation, maximum, minimum, multiplication, and subtraction constructed using workflow illustrated in Figure 8.0.This process involves feeding the input data (either 20 or 40 features) into both the shallow and deep models. The models then process the data through their respective hidden layers.

The outputs of the shallow and deep models are subsequently combined using various fusion functions, and the final output is produced using a sigmoid function. In the workflow, the input features (shape: None, n) are simultaneously fed into both the Shallow and Deep models. The 'None' parameter allows for a flexible batch size, enabling the models to process varying numbers of input samples. The value 'n' represents the number of features per input sample. Both models process the input data and produce predicted tensors with a shape of (None, 1). These tensors are then combined into a single prediction tensor, also of shape (None, 1), using a decision fusion function that applies element-wise operations. For example, the maximum function selects the maximum value from the two input tensors.

Finally, this result is passed through an output dense layer that applies a linear transformation defined by

$$y = Wz + b$$

using the sigmoid activation function given by

$$f(x) = \frac{1}{1 + e^{-x}}$$

to produce the final prediction of the process[35].This approach led to the creation of 12 hybrid models, as illustrated in Figure 6 [35].

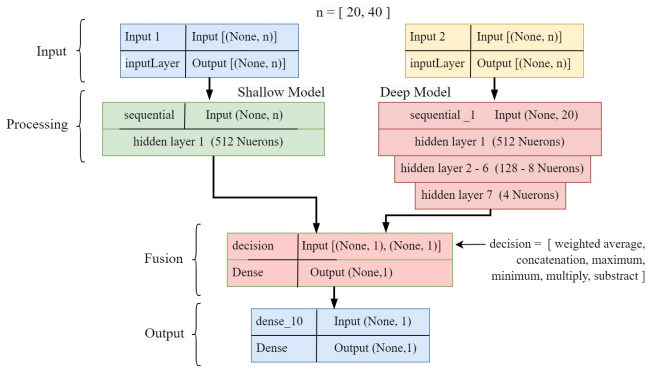

Fig. 6 Hybrid fusion model creation workflow

The study defines the components involved in the decision-level fusion process as follows:

- Let x represent the input data.
- Let S_p be the prediction from the shallow model.
- Let D_p be the prediction from the deep model.
- Let z represent the combined prediction obtained through the fusion method.
- Let $h(z)$ be the processing function applied to the combined predictions.
- Let $\hat{y}$ be the final prediction using the sigmoid output layer.

Shallow model prediction

The function f_S computes the prediction S_p based on the input data x using the shallow model.

$$S_p = f_S(x)$$

Deep model prediction

Similarly, f_D computes the prediction D_p based on the same input data x but using the deep model.

$$D_p = f_D(x)$$

Fusion of predictions

The fusion function combines the predictions S_p and D_p to produce a unified output z.

$$z = \text{FusionFunction}(S_p, D_p)$$

The fusion function can take several forms, including weighted averaging, concatenation, maximum, minimum, multiplication, and subtraction, as described in Table 4.

Final prediction

The sigmoid function is applied to the output z of the fusion function to produce the final prediction $\hat{y}$, which ranges between 0 and 1, making it suitable for binary classification tasks.

$$\hat{y} = \text{sigmoid}(z)$$

Fusion functions

The individual fusion function descriptions and formals are given below. Let y represent the final fused output as shown in Table 4.

Table 4 Fusion functions and formulas

Fusion Function	Description	Formula
Weighted Averaging	Computes the weighted sum of the outputs of two models. The weights are predefined based on the complexity of each model. This study predefined weights of 0.15 and 0.85 on the complexity of Shallow and Deep models based on the number of hidden layers (1:7).	$y = Z(w_1 \cdot S_p + w_2 \cdot D_p)$ where w_1 and w_2 are predefined weights.
Concatenation	Combines the outputs from multiple models by joining them end-to-end along a specified dimension.	$y = \text{Concatenate}(S_p, D_p)$
Maximum	Obtains the final output by taking the element-wise maximum of outputs from the models.	$y = \text{maximum}(S_p, D_p)$
Minimum	Obtains the final output by taking the element-wise minimum of the outputs from the models.	$y = \text{minimum}(y_1, y_2)$
Multiply	Obtains the final output by taking the element-wise product of the outputs from the models.	$y = (y_1 \cdot y_2)$
Subtract	Obtains the final output by taking the element-wise difference of the outputs from the models.	$y = (y_1 - y_2)$

The 12 resulting hybrid fusion neural network models, created by combining the Shallow20-Deep20 and Shallow40-Deep40 models using the five fusion functions described in Table 4, are illustrated in Figure 7.

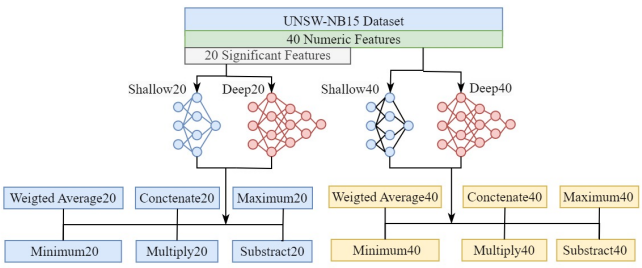

Fig. 7 Architectures of 12 hybrid fusion models

3.7 Training, testing and evaluation

The study trained 12 Hybrid Fusion NN models using 20-feature and 40-feature datasets. Training performance was monitored in real-time with a validation dataset to prevent overfitting. Early stopping with a patience of 50 was used to save the model at the epoch with the lowest validation loss. This approach mitigates overfitting and underfitting, ensuring optimal performance on unseen data. Performance metrics from training and testing demonstrated the models' effectiveness in real-world scenarios, helping to identify the best [39].

3.7.1 Performance metrics

The study evaluated performance using various metrics, including validation loss, validation accuracy, test accuracy, precision, recall, F1-score, model size, CPU usage, and memory usage. Validation accuracy and loss curves were analyzed to assess the model's generalization, pattern detection, and prediction capabilities during training. The predictive performance of the models was evaluated using true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) derived from the confusion matrix on the test dataset. These metrics were used to calculate key performance indicators, including accuracy, precision, recall (sensitivity), and F1 score, to evaluate the model's effectiveness.

Accuracy provides a measure of overall performance. Precision provides a measure of the accuracy of positive predictions. Recall provides a measure of the ability of a model to find all the correctly predicted instances. The F1 Score provides the harmonic mean of precision and recall [41]. The study obtained the ROC Curve and AUC Score using test data to evaluate the model performance on unseen instances. This was done by plotting the true positive rate (TPR) against the false positive rate (FPR) at various classification thresholds [39]. Each model was compiled into an executable and tested in a controlled environment, measuring the time to predict 9,300 test samples. This metric is crucial for assessing real-time performance and scalability. CPU and memory usage were also monitored to evaluate resource utilization. These metrics informed the decision on the best model for deployment on specific edge servers.
The deployment and monitoring stages of the ML pipeline used for real-world applications were omitted in this study, as the study objective was model development and evaluation only.

4 Results and discussion

4.1 Model performance

Tables 5 and 6 present the training and testing results for the six models using the 20-feature dataset with show model, end epoch (ee), training time (M) (tt) ,

validation loss (vl), test accuracy (ta), precision (pr), recall (rc), f1 score (f1), roc Score (rs), and prediction time (M) (pt) columns.

Table 5 20-feature model performance (Part 1)

Model	ee	tt	vl	ta	pr
Shallow20	14	7.28	0.17	0.93	0.95
Deep20	128	29.54	0.12	0.95	0.97
WgtAv20	217	74.23	0.12	0.95	0.96
Concat20	113	40.89	0.13	0.95	0.96
Minimum20	142	60.09	0.13	0.95	0.97
Maximum20	190	48.71	0.13	0.95	0.97
Multiply20	165	48.69	0.13	0.95	0.96
Subtract20	174	71.89	0.13	0.95	0.96

Table 6 20-feature model performance (Part 2)

Model	recall	f1	rs	pt
Shallow20	0.91	0.93	0.99	0.52
Deep20	0.93	0.95	0.99	0.62
WgtAv20	0.94	0.95	0.99	0.71
Concat20	0.93	0.95	0.99	0.86
Minimum20	0.93	0.95	0.99	0.71
Maximum20	0.93	0.95	0.99	0.71
Multiply20	0.93	0.95	0.99	0.69
Subtract20	0.94	0.95	0.99	0.99

Figure 8 shows the live monitoring of Training accuracy for 20 feature fusion models during training.

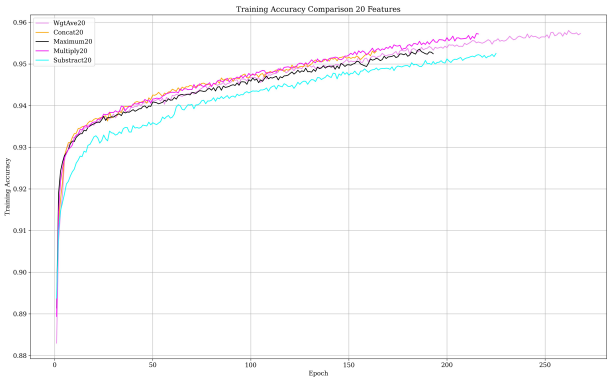

Fig. 8 Training accuracy of 20-feature models

Figure 9 shows the live monitoring of validation loss for 20 feature fusion models during training.

The training and testing results of 6 models employing a 40-feature dataset are shown in Tables 7 and 8.

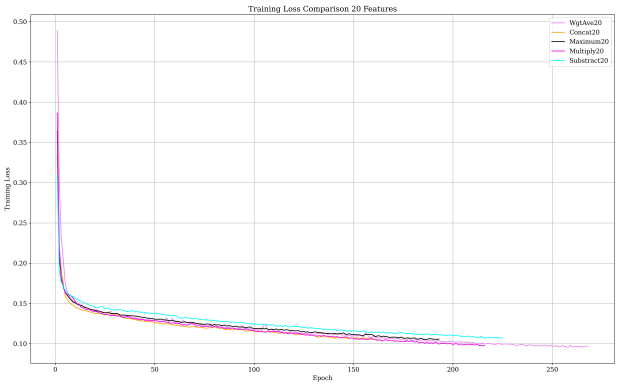

Fig. 9 Validation loss of 20-feature models

Table 7 40-Feature model performance (Part 1)

Model	ee	tt	vl	ta	pr
Shallow40	61	15.0	56.0	0.94	0.96
Deep40	68	18.5	59.0	0.97	0.97
WgtAv40	90	28.86	0.05	0.98	0.98
Concat40	50	18.91	0.05	0.98	0.98
Minimum40	81	24.88	0.06	0.98	0.99
Maximum40	74	38.26	0.05	0.98	0.98
Multiply40	131	57.81	0.06	0.98	0.99
Subtract40	141	70.18	0.06	0.98	0.98

Table 8 40-feature fusion model performance (Part 2)

Model	recall	f1	rs	pt
Shallow40	0.92	0.94	0.99	0.56
Deep40	0.97	0.97	0.99	0.59
WgtAv40	0.98	0.98	1.00	0.82
Concat40	0.98	0.98	1.00	0.71
Minimum40	0.98	0.98	1.00	0.63
Maximum40	0.98	0.98	1.00	0.73
Multiply40	0.97	0.98	1.00	1.29
Subtract40	0.98	0.98	1.00	1.17

Figure 10 shows the live monitoring of accuracy for 40-feature models during training.

Figure 11 shows the live monitoring of validation loss for 40-feature models during training. The training was stopped when validation loss ceased to reduce further to prevent overfitting.

4.1.1 Training time

The individual shallow model, trained on 186,000 samples split 90:5:5 for training, testing, and validation, achieved the fastest training time of 7.28 minutes and converged in 14 epochs. In contrast, despite a quicker initial training time of 29.54 minutes, the Deep mode required 128 epochs to converge. Hybrid fusion models had longer training times and more epochs, with the

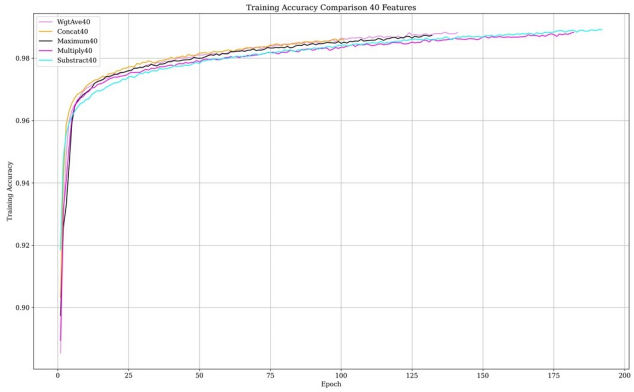

Fig. 10 Training accuracy of 40-feature model

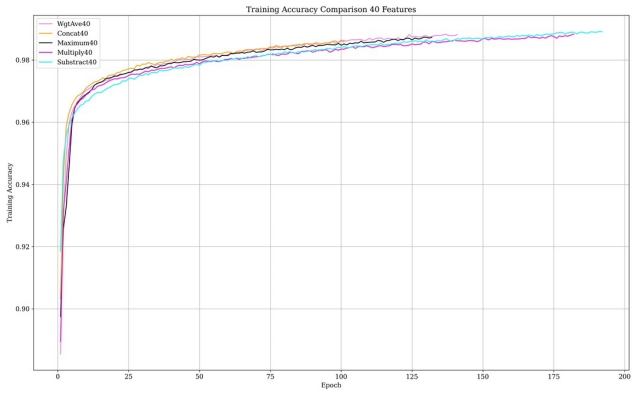

Fig. 11 Validation loss and training accuracy of 40-feature models

Subtract40 model taking the longest at 70.18 minutes, as shown in Figure 12.

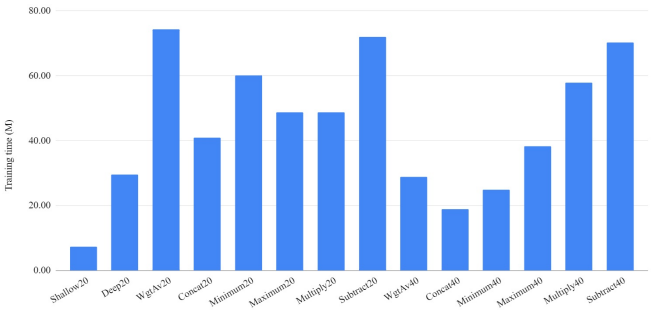

Fig. 12 Training time of hybrid fusion models

4.1.2 Validation loss

Higher validation loss can negatively affect federated and one-time learning by causing poor model generalization, aggregation issues, and slower convergence [41]. It may also indicate overfitting, leading to poor performance on edge devices. The 40-feature models achieved a lower validation loss (ranging from 0.05 to 0.06) compared to the 20-feature models (ranging from 0.12 to 0.17). Therefore, Maximum40 and Concat40 hybrid models are more suitable for federated and one-time learning based on validation loss.

4.1.3 Prediction time

Prediction time is crucial for evaluating NN models used in edge servers, as it impacts real-time processing, resource utilization, and scalability [41]. Figure 17 displays the prediction time for 9,300 samples across all models as shown in Figure 13.

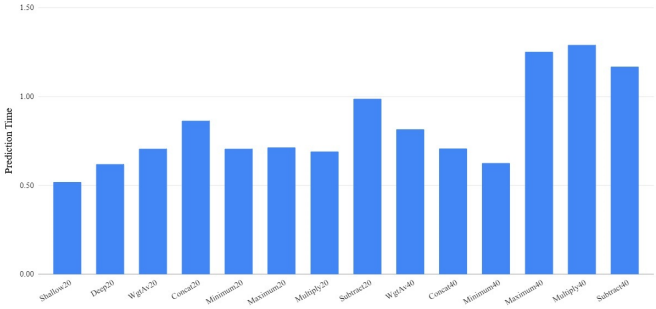

Fig. 13 Prediction time of hybrid fusion models

The Shallow20 model had the lowest prediction time at 0.52 seconds for 9,300 samples, while the Concat40 and Minimum40 hybrid models took 0.63 and 0.71 seconds, respectively. Although these differences may seem minor, they can lead to significant response delays in scaled edge environments, such as autonomous vehicles, where rapid detection of attacks from large traffic samples is crucial.

4.2 Radar analysis

The radar comparison used a ranking strategy, assigning ranks from 6 of the 20-feature hybrid fusion model metrics. When assigning rankings, lower rankings were given for better performance for epoch, training time, validation loss, and prediction time. Higher rankings were given for better performance for test accuracy, precision, recall, F1 score, and ROC AUC score. The final rank summing up the individual ranks. The overall ranks of 20-feature fusion models are given in Table 9.

Figure 14 displays a radar chart comparing key performance metrics of 6 Hybrid Fusion models using 20 features. The metrics include end epoch, training time, validation loss, testing time, test accuracy, precision, recall, and F1 score. All results are normalized using min-max scaling, ranging each metric between 0 and

Table 9 40-Feature model ranking for radar comparison

Model	Calculation	Rank
Maximum20	$3+2+3+1+1+3+1+1+2$	17
Concat20	$1+1+2+1+3+3+1+1+5$	18
Minimum20	$2+4+3+1+1+3+1+1+2$	18
Multiply20	$4+3+3+1+3+3+1+1+1$	20
WgtAv20	$6+6+1+1+3+1+1+1+4$	24
Subtract20	$5+5+3+1+3+1+1+1+6$	26

Table 10 40-Feature model ranking for radar comparison

Model	Calculation	Rank
Concat40	$1+1+1+1+3+1+1+1+2$	12
WgtAv40	$2+2+1+1+3+1+1+1+3$	15
Minimum40	$4+3+4+1+1+1+1+1+1$	17
Maximum40	$3+4+1+1+3+1+1+1+5$	20
Subtract40	$6+6+5+1+3+1+1+1+4$	28
Multiply40	$5+5+5+1+1+6+1+1+6$	31

1. Based on these calculations, Maximum20 was the best-performing model, followed by Concat20 and Minimum20.

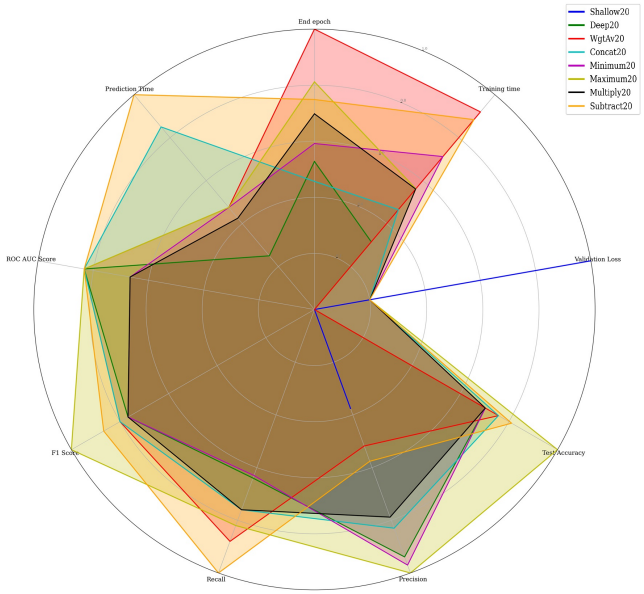

Fig. 14 Key performance metrics of 20 feature models

Based on these calculations, Maximum20 was the best-performing model, followed by Concat20 and Minimum20. Figure 19 displays a radar chart comparing key performance metrics of 6 models using 40 features. Using a similar ranking approach, overall ranks for 40-feature fusion models were calculated as given in Table 10.

Figure 15 displays a radar chart comparing key performance metrics of 6 Hybrid Fusion models using 40 features. Concat40 performed best, followed by WgtAve40 and Minimum40. Among both 20-feature and 40-feature Hybrid Fusion models, Concat40 was the top performer.

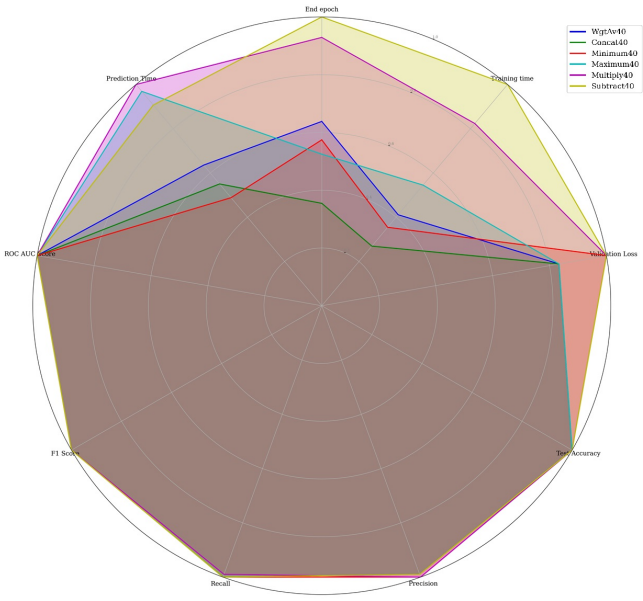

Fig. 15 Key performance metrics of 40 feature models

4.3 Resource usage

The study was conducted in a controlled environment using dedicated hardware to ensure consistent conditions for evaluating resource usage. Testing was performed on multi-core processors, simulating realistic, resource-intensive conditions. The models were compiled into standalone executables using PyInstaller, allowing them to run independently of their development environment. CPU and Memory resource usage was monitored in real-time using Windows Resource Monitor. The models were evaluated in a batch prediction scenario, where multiple predictions were processed simultaneously. The results are shown in Table VII. The CPU usage reflects the computational load during prediction or training. Memory usage reflects RAM consumption during model loading, batch prediction, and unloading.

The columns in the Table 11 denotes Shallow20 (s20), Deep20 (d20), Shallow40 (s40), Deep40 (d40), Concat20 (c20).

Table 11 Resource utilization by models (Part 1)

Metric	s20	d20	s40	d40	c20
Prediction Time (S)	0.65	0.82	0.47	0.54	0.70
Max CPU (%)	0.30	0.27	0.34	0.33	0.19
Memory Allocated (GB)	28.12	28.00	31.44	29.78	29.06
Memory Used (MB)	19.91	19.73	23.13	21.59	20.78
Model Size (KB)	110	648	190	708	794

The columns in the Table 12 denotes Concat40 (c40), Maximum20 (mx20), Maximum40 (mx40), Minimum20 (mn20) and Minimum40 (Mn40).

Table 12 Resource utilization by models (Part 2)

Metric	c40	mx20	mx40	mn20	mn40
Prediction Time (S)	0.86	0.70	0.73	0.64	0.70
Max CPU (%)	0.22	0.26	0.31	0.29	0.36
Memory Allocated (GB)	31.93	29.02	30.39	30.85	31.32
Memory Used (MB)	23.60	20.27	22.22	21.57	23.08
Model Size (KB)	794	788	1086	788	968

Figure 16 presents a radar chart that compares key resource utilization metrics for two individual models and six Hybrid Fusion models. The chart provides a visual representation of how each model performs across multiple resource categories, such as CPU usage, memory usage, and model size, allowing for easy comparison of resource efficiency between the models.

Based on the radar chart in Figure 16, the Shallow20 model has the lowest resource usage among the 20-feature models. For 40-feature models, Maximum40 is the most resource-efficient. Shallow20 has the smallest model size with the lowest inference time, and moderate memory and CPU usage, indicating good resource efficiency. Concat20 and Maximum40, despite their larger model sizes (794 KB and 788 KB respectively) compared to Shallow20 (110 KB), are competitive in CPU and memory usage. Concat20 offers the lowest memory and CPU usage at 20.78 MB and 0.19%, respectively. For low-resource edge environments, prioritizing smaller model sizes and lower CPU and memory usage while maintaining acceptable accuracy is beneficial. Shallow20 offering acceptable performance and low resource usage, is suitable for such environments. In high-resource settings, where accuracy is prioritized even with larger model sizes and resource usage, Concat20, Maximum40, and Minimum40 are appropriate choices.

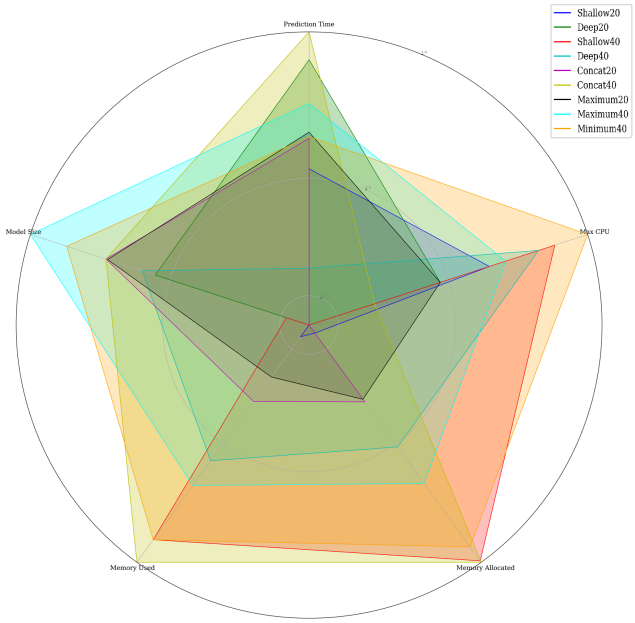

Fig. 16 Resource utilization results of selected models

4.4 Correlation analysis

Correlation analysis, shown in Figure 17, provides insights into the relationships between factors and performance metrics. It helps understand model behavior, detect potential overfitting, compare model variations, and validate assumptions [41]. Ultimately, it aids in selecting better models and optimizing their configurations for improved performance. The ROC AUC score

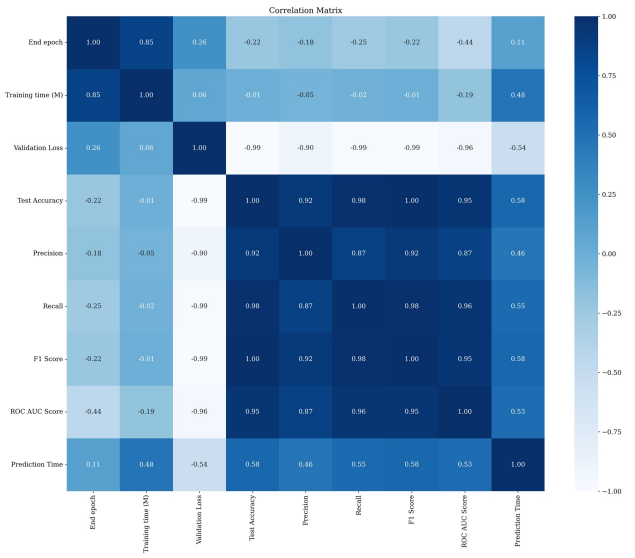

Fig. 17 Correlation analysis of evaluation metrics

has a moderate negative correlation of -0.44 with the number of epochs, indicating it tends to decrease with more epochs. Test accuracy, precision, recall, F1 score, and ROC AUC score show strong negative correlations ranging from -0.89 to -0.99 with validation loss, meaning these metrics improve significantly as validation loss decreases. These performance metrics are highly positively correlated (0.87 to 1.00), indicating they move together. A moderate positive correlation of 0.48 exists between prediction time and training time, suggesting longer training times may lead to longer prediction times. Moderate negative correlation of -0.54 between prediction time and validation loss, suggests that models with longer prediction times may have lower validation losses. This analysis indicates that faster converging models generally offer better performance and lower overfitting, supporting the suitability of Concat40, Minimum40, and Maximum40 for high performance and reduced overfitting in edge server network attack detection. The evaluation reveals that hybrid models generally outperform individual Shallow and Deep models in accuracy, precision, recall, F1 score, and ROC AUC score, demonstrating their robust performance in detecting network attacks on edge servers. Although hybrid models have longer training times more epochs to converge, and slightly higher inference times, Concat40, Maximum40, and Minimum40 offer the best balance of performance and resource usage. Specifically, Concat40 is preferable for its computational efficiency and lower resource usage, making it suitable for federated learning tasks. Maximum20 and Maximum40 are effective for one-time training tasks. Overall, hybrid models, show significant improvements over individual models, enhancing network attack detection on edge servers.

4.5 Future directions

This study aims to further explore communication security in autonomous vehicles by refining and optimizing hybrid models, to address the dynamic nature of vehicular networks for enhancing the security and reliability of AVs.

5 Conclusion

This research demonstrates the effectiveness in applicability of Shallow-Deep Hybrid Fusion models for detecting network attacks on edge servers. The results show that Hybrid Fusion models outperform individual Shallow and Deep models across key performance metrics. Among the Hybrid Fusion models experimented with the Maximum20 model, achieving an accuracy of 95.34

The study recognizes the trade-offs between model performance and resource consumption. In this context, Hybrid Fusion models using 40 features demonstrate higher accuracy and better performance metrics, but they require more computational resources and longer training times. The Concat40 and Maximum40 models demonstrated an overall balance between performance and resource efficiency. Hence, make them ideal models for deployment in high-resource edge environments for network attack detection. The study noted the Shallow20 model's significant resource efficiency, smaller model size, lowest inference time, and low memory and CPU usage. Although it achieved a moderate accuracy of 0.93, the Shallow20 model makes it a viable option for ultra-low-resource edge environments demanding low latency over accuracy as the first line of defense, where quick deployment and efficient resource utilization are critical.

Overall, the Shallow-Deep Hybrid Fusion models offer a robust solution for network attack detection on edge servers. They are capable of adapting to varying resource constraints. These findings offer valuable insights to optimize edge server security. as well as show the potential for applying them beyond cybersecurity to other domains. Future research aims to explore communication security in autonomous vehicles, prioritizing low latency, high reliability, and security by optimizing hybrid models to ensure the safety of autonomous vehicles.

References

1. M. Ahmed and P. H. Dowland, *Secure Edge Computing*. CRC Press, 2021.
2. M. Satyanarayanan, "Edge Computing," *Computer*, vol. 50, no. 10, pp. 36–38, 2017.
3. Gartner, "What Edge Computing Means for Infrastructure and Operations Leaders," 2018.
4. E. Fazeldehkordi and T.-M. Grønli, "A Survey of Security Architectures for Edge Computing-Based IoT," *IoT*, vol. 3, no. 3, pp. 332–365, 2022.
5. J. Cook, S. U. Rehman, and M. A. Khan, "Security and Privacy for Low Power IoT Devices on 5G and beyond Networks: Challenges and Future Directions," *IEEE Access*, vol. 11, pp. 39295–39317, 2023.
6. Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv, "Edge Computing Security: State of the Art and Challenges," *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1608–1631, 2019.
7. N. W. Meegammana and H. Fernando, "Detection of Network Attacks on Application Servers Using Deep Learning in IoT Environments," 2023. [Online]. Available: `https://doi.org/10.1109/icac60630.2023.10417159`.
8. 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.

9. X. Jin, C. Katsis, F. Sang, J. Sun, A. Kundu, and R. R. Kompella, "Edge Security: Challenges and Issues," *arXiv*, 2022. [Online]. Available: https://doi.org/10.48550/arxiv.2206.07164.
10. S. Jasper, *Russian Cyber Operations*. Georgetown University Press, 2022.
11. U. Tariq, I. Ahmed, A. K. Bashir, and K. Shaukat, "A Critical Cybersecurity Analysis and Future Research Directions for the Internet of Things: A Comprehensive Review," *Sensors*, vol. 23, no. 8, 2023.
12. 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.
13. Y. Shi, H. Feng, X. Geng, X. Tang, and Y. Wang, "A Survey of Hybrid Deep Learning Methods for Traffic Flow Prediction," 2019. [Online]. Available: https://doi.org/10.1145/3373419.3373429.
14. A. Sagu, N. S. Gill, and P. Gulia, "Hybrid Deep Neural Network Model for Detection of Security Attacks in IoT Enabled Environment," *International Journal of Advanced Computer Science and Applications*, vol. 13, no. 1, 2022.
15. N. W. Meegammana and H. Fernando, "Securing IoT Servers: Shallow vs. Deep Neural Network Architectures," Unpublished, 2024.
16. T. Qiu, J. Chi, X. Zhou, Z. Ning, M. Atiquzzaman, and D. O. Wu, "Edge Computing in Industrial Internet of Things: Architecture, Advances and Challenges," *IEEE Communications Surveys & Tutorials*, 2020.
17. C. C. Aggarwal, *Neural Networks and Deep Learning: a Textbook*. Springer International Publishing, 2018.
18. N. Moustafa, M. Abdel-Basset, R. Mohamed, *Deep Learning Approaches for Security Threats in IoT Environments*. Wiley-IEEE Press (2022)
19. L. Caviglione, C. Comito, M. Guarascio, G. Manco, Emerging Challenges and Perspectives in Deep Learning Model Security: a Brief Survey, Systems and Soft Computing **2023** (2023). doi:10.1016/j.sasc.2023.200050
20. 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). Accessed: 2024-04-18
21. H. G. Abreha, M. Hayajneh, M. A. Serhani, Federated Learning in Edge Computing: a Systematic Survey, Sensors **22**(2) (2022). doi:10.3390/s22020450
22. N. A. Sulieman, L. Ricciardi Celsi, W. Li, A. Zomaya, M. Villari, Edge-Oriented Computing: a Survey on Research and Use Cases, Energies **15**(2) (2022). doi:10.3390/en15020452
23. H. Zeyu, X. Geming, W. Zhaohang, Y. Sen, Survey on Edge Computing Security, https://ieeexplore.ieee.org/abstract/document/9196442 (2020)
24. H. Kim, K. Lee, IIoT Malware Detection Using Edge Computing and Deep Learning for Cybersecurity in Smart Factories, Applied Sciences **12**(15) (2022). doi:10.3390/app12157679
25. O. Jullian, B. Otero, E. Rodriguez, N. Gutierrez, H. Antona, R. Canal, Deep-Learning Based Detection for Cyber-Attacks in IoT Networks: A Distributed Attack Detection Framework, Journal of Network and Systems Management **31**(2) (2023). doi:10.1007/s10922-023-09722-7
26. Y. Wu, D. Wei, J. Feng, Network Attacks Detection Methods Based on Deep Learning Techniques: a Survey, Security & Communication Networks, pp. 1-17 (2020-08). doi:10.1155/2020/8872923
27. Y. Zhang, Y. Liu, X. Guo, Z. Liu, X. Zhang, K. Liang, A BiLSTM-Based DDoS Attack Detection Method for Edge Computing, Energies **15**(21), 7882 (2022). doi:10.3390/en15217882
28. M. A. Ganaie, M. Hu, A. K. Malik, M. Tanveer, P. N. Suganthan, Ensemble Deep Learning: a Review, Engineering Applications of Artificial Intelligence **115**, 105151 (2022). doi:10.1016/j.engappai.2022.105151
29. S. Abimannan, E.-S. M. El-Alfy, Y.-S. Chang, S. Hussain, S. Shukla, D. Satheesh, Ensemble Multifeatured Deep Learning Models and Applications: a Survey, IEEE Access **11**, 107194-107217 (2023). doi:10.1109/access.2023.3320042
30. Y. Liu, N. Kumar, Z. Xiong, W. M. Lim, R. Yu, D. Niyato, Communication-Efficient Federated Learning for Anomaly Detection in Industrial Internet of Things, IEEE Globecom 2020, (2020). doi:10.1109/globecom42002.2020.9348249
31. S. Ullah, et al., HDL-IDS: a Hybrid Deep Learning Architecture for Intrusion Detection in the Internet of Vehicles, Sensors **22**(4), 1340 (2022). doi:10.3390/s22041340
32. E.-H. Qazi, M. H. Faheem, T. Zia, HDLNIDS: Hybrid Deep-Learning-Based Network Intrusion Detection System, Applied Sciences **13**(8), 4921 (2023). doi:10.3390/app13084921
33. D. Javeed, T. Gao, M. T. Khan, I. Ahmad, A Hybrid Deep Learning-Driven SDN Enabled Mechanism for Secure Communication in Internet of Things (IoT), Sensors **21**(14), 4884 (2021). doi:10.3390/s21144884
34. M. A. Khan, et al., A Hybrid Deep Learning-based Intrusion Detection System for IoT Networks, Mathematical Biosciences and Engineering **20**(8), 13491-13520 (2023). doi:10.3934/mbe.2023602
35. W. Li, Y. Peng, M. Zhang, L. Ding, H. Hu, L. Shen, Deep Model Fusion: a Survey, arXiv (Cornell University) (2023). doi:10.48550/arxiv.2309.15698
36. H. Sedjelmaci, S. M. Senouci, N. Ansari, A. Boualouache, A Trusted Hybrid Learning Approach to Secure Edge Computing, IEEE Consumer Electronics Magazine **2021** (2021). doi:10.1109/mce.2021.3099634
37. H. Hapke, C. Nelson, *Building Machine Learning Pipelines: Automating Model Life Cycles with Tensorflow*. O'Reilly, Beijing; Boston; Farnham; Sebastopol; Tokyo (2020)
38. W. David, UNSW_NB15 Dataset, https://www.kaggle.com/datasets/mrwellsdavid/unsw-nb15 (2019). Accessed: Feb. 25, 2024
39. I. Goodfellow, Y. Bengio, A. Courville, *Deep Learning*. MIT Press (2016)
40. D. Y. Choi, D.-H. Kim, B. C. Song, Multimodal Attention Network for Continuous-Time Emotion Recognition Using Video and EEG Signals, IEEE Access **8**, 203814-203826 (2020). doi:10.1109/access.2020.3036877
41. F. Chollet, *Deep Learning with Python*. Manning, Shelter Island, New York (2018)