**Leveraging Shallow-Deep Hybrid Neural Networks for Optimizing Attack Detection on Edge Servers**

Niranjan W. Meegammana, Harinda Fernando

Shilpa Sayura Foundation, Sri Lanka

Sri Lanka Institute of Information Technology, Sri Lanka

[niranjan.meegammana@gmail.com](mailto:niranjan.meegammana@gmail.com), [harinda.f@sliit.lk](mailto:harinda.f@sliit.lk)



*Abstract****— This study investigates Shallow-Deep Hybrid Fusion Neural Networks (NNs) for detecting network attacks on edge servers. It follows a systematic machine learning workflow to create and evaluate multiple Shallow-Deep Hybrid Fusion models, employing parallel fusion techniques for attack classification. The research combined a single-layer Shallow model with 512 neurons, and a seven-layer Deep model with varying neuron counts to construct 12 Shallow-Deep Hybrid Fusion models by employing weighted sum, concatenation, minimum, maximum, multiplication, and subtraction fusion functions. The hybrid models were trained, validated, and tested using 20 and 40 feature datasets derived from the UNSW-NB15 dataset, and evaluated on performance metrics along with model size, prediction time, memory, and CPU usage.***

***The results suggest that 20-feature maximum and minimum hybrid fusion models achieved 95.00% accuracy and a precision score of 0.97, applicable for resource-constrained edge servers. The 40-feature concat and maximum hybrid fusion models achieved 98.00% accuracy and a precision score of 0.99, outperforming other models, indicating their suitability for high-resource edge servers. The study proposes exploring the 20-feature hybrid fusion maximum model for federated learning in low-resource edge servers.***

***In conclusion, Shallow-Deep Hybrid Fusion models are a preferable choice for detecting network attacks on edge servers, and suitable for both high and low-end edge servers. This research contributes valuable insights into enhancing the security of edge servers. Future research will focus on employing these hybrid models with federated learning for securing autonomous vehicle environments. Ultimately, the study suggests the application of Shallow-Deep Hybrid Fusion models beyond cyber security across diverse domains and applications.***

***Keywords— Shallow-Deep, Parallel Fusion, Hybrid Neural Networks, Attack Detection, Edge Servers***

1. **Introduction**

This Edge network architecture facilitates distributed computing by positioning data processing resources close to data sources. It aims to ensure low latency, reduced bandwidth, and high reliability for edge applications. Despite these advantages, edge servers face unique network security challenges that threaten their integrity. Neural Network techniques offer promising capabilities to mitigate these challenges (Ahmed and Dowland, 2021). This research investigates the application of a Shallow-Deep Hybrid Fusion Neural Network solution to mitigate network attacks on edge servers. Figure 1 illustrates the architecture of an edge environment.

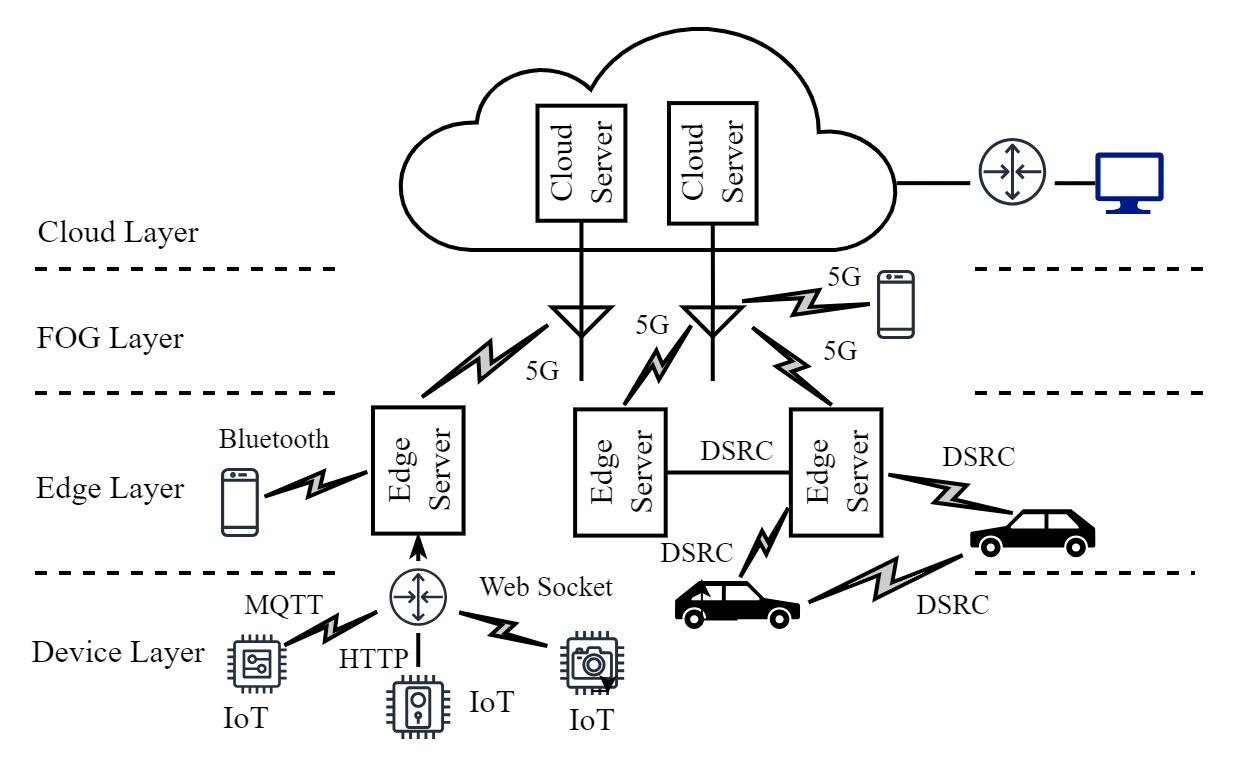


Fig. 1 Architecture of an edge environment

Edge Computing Environments

Evolved with the advent of the Internet of Things (IoT), edge computing offers unparalleled advantages by decentralizing crucial computational resources, like application servers, near IoT devices, sensors, mobiles, and users. An edge server located locally connects and controls IoT devices facilitating data collection, aggregation, processing, and communicating with the cloud. The edge architecture improves latency, reduces network bandwidth while increasing privacy, and harnesses survivability (Satyanarayanan, 2017). Gartner projects that over 75 percent of enterprise data will be generated and processed outside the cloud by 2025, due to the rapid expansion of edge infrastructures across various domains (Gartner, 2018). Figure 1 illustrates the architecture of an edge environment.

Edge Network Security

The rapid proliferation of edge networks poses significant challenges to edge applications and data security. Adversaries exploiting inherent vulnerabilities in IoT devices employed in edge networks can carry out a range of malicious activities, including lateral attacks, DDoS attacks, data exfiltration, and injection attacks, including identity theft, among others. For example, lateral attacks allow moving horizontally across an edge network from one device to another, then leveraging weak security measures to gain unauthorized access to mission-critical edge servers. Unsecured IoT devices on edge networks enable orchestrating large-scale DDoS attacks to flood edge servers with excessive network traffic to make them unavailable to legitimate users. Data exfiltration targets edge servers to steal sensitive data for malicious purposes, such as espionage, fraud, extortion, or identity theft. Injection attacks involve inserting malicious commands or code into edge servers by exploiting software and protocols employed (Fazeldehkordi and Grønli,2022). For instance, SQL injection attacks target edge servers with web interfaces. An edge server allowing users to execute certain commands without properly sanitizing user input opens avenues for attackers to inject malicious commands to delete important system files or data.

Malicious actors frequently recruit edge devices to form botnets to carry out spam campaigns, and crypto-jacking The rise of 5G networks further heightens security concerns within edge environments. Safeguarding edge servers from cyber attacks is challenging due to limited processing power, extensive distribution, and the absence of universal security standards. Conversational security mechanisms such as IDS/IPS face difficulties in protecting edge networks due to their distinctive architecture, which is significantly different from traditional network structures (Cook, Rehman, and Khan, 2023). The heterogeneity of edge environments created by diverse devices and platforms. It complicates the standardization of security protocols, exposing devices vulnerable to known exploits. Mitigating these risks necessitates a comprehensive approach to protect infrastructure, applications, devices, and sensitive data within edge environments (Xiao et al., 2019).

Impact of Edge Security Breaches

Breaches of edge networks can result in significant disruptions to critical services, compromise vital information cause financial losses, and pose enormous risks to critical operations (Meegammana and Fernando, 2023). Well-known examples are the Mirai botnet attack in 2016 (Xiao et al., 2019), and DDoS attacks on Spotify. Twitter, and Netflix attacks in 2016 were traced back to compromised IoT devices (Alwarafy et al., 2020). Edge servers, managing and controlling a vast amount of sensitive information, are increasingly becoming prime targets for cyber attacks initiated through connected IoT devices. Hence, the security of edge servers is a significant concern in ensuring the delivery of critical services, where such breaches not only disrupt essential operations but also incur substantial financial losses and jeopardize public safety. Moreover, the rapid evolution of attacks, coupled with the heterogeneity of edge environments, exacerbates the difficulty of mitigating these risks effectively. Compromised edge devices in healthcare and automotive sectors can escalate into life-threatening situations. Exploitation of edge servers monitoring the environment can lead to devastating consequences (Jin et al., 2022).

Additionally, these security breaches can cause remediation costs, legal fees, regulatory fines, and loss of revenue, including damage to brand reputation. Importantly, high-profile breaches on edge networks can erode trust among stakeholders and undermine confidence in the security practices of organizations. Organizations subject to regulatory requirements such as GDPR and HIPAA concerning data protection and privacy face the risk of non-compliance, leading to legal consequences and penalties. Heightened geopolitical tensions, such as the Russia-Ukraine conflict, may incentivize malicious entities to breach edge servers within critical infrastructure, leading to significant disruptions in utility services and potentially sparking cyber warfare (Jasper, 2020). Considering these critical risks posed by security breaches in edge environments, organizations need to prioritize robust security measures to ensure mitigation of the risks associated with edge security breaches to ensure the integrity and continuity of services, particularly within the critical infrastructure to ensure the delivery of services for citizens, to uphold societal functioning, and shield economies and societies from crises (Tariq et al., 2023).

Challenges of Securing in Edge Environments

Traditional edge server security measures such as firewalls, anti-virus solutions, and IDS face several challenges in effectively addressing the evolving cyber threats due to the unique characteristics of edge networks that are different from conventional networks. Edge devices have limited processing power, memory, and energy resources. Hence, it makes it challenging to implement robust security measures such as encryption, authentication, and intrusion detection without compromising their performance. Edge environments encompassing heterogeneous devices and platforms from various vendors. TThesediverse security capabilities and configurations complicate employing standardized security measures, requiring device-type-based security strategies. The lack of standardization in security protocols and practices across edge devices and networks hinders interoperability, making it difficult to enforce consistent security policies and controls (Meegammana and Fernando, 2023).

Neural Networks for Edge Security

Research shows that Neural Networks (NN) based network attack detection systems enable organizations to detect and respond to cyber threats in real-time. NNs extend conventional machine learning solutions effectively in the cybersecurity domain by handling critical tasks such as attack classification and malware pattern recognition with high precision (Sagu, Gill, and Gulia, 2020). The advantage of NN models is that their layered architecture offers flexibility. NN models can capture instinct patterns in network traffic data to address diverse problems such as anomaly detection, intrusion detection, threat intelligence, behavioral analysis, privacy preservation, and adaptive defense providing real-time response in edge environments. Feed Forward Neural Networks (FFN) constructed using one or more hidden layers together with input and output layers demonstrate remarkable effectiveness for successfully detecting network attacks on IoT application servers achieving notable high accuracy levels with 99 percent precision. (Meegammana and Fernando , 2023).

Hybrid Neural Network Models

Hybrid NNs constructed by combining multiple models allow for gaining advanced problem-solving capabilities. They leverage various model combinations by addressing weaknesses in one model with the strengths of the other model. Hybrid models capitalize on their unique model strengths, to help improve overall predictive performance, reduce overfitting, and increase robustness and resilience. Hybrid NN models offer greater flexibility in designing various NN architectures to increase model interpretability. Conventional hybrid techniques such as bagging, stacking, and boosting, as well as newer weighted average, concatenation, and multiplication approaches, are integral to these versatile NN architectures (Shi et al., 2019). Numerous studies have explored the applications of hybrid NN models in diverse domains (Sagu, Gill, and Gulia, 2022), (Meegammana and Fernando, 2024).   
  
However, there is a scarcity of research on the utilization of Shallow-Deep Hybrid Fusion approaches in cyber security. Hence, research on Shallow-Deep Hybrid Fusion architectures has the potential to offer promising solutions for enhancing the security of edge application servers to safeguard against constantly evolving sophisticated cyber threats. This research aims to extend existing knowledge in fortifying edge servers, by investigating the applicability of a Shallow-Deep Hybrid Fusion architecture in combating potential network attacks against edge application servers.

This paper is structured into five sections. It commences with this introduction to the research context followed by a review of existing literature, analyzing previous studies on security risks in edge environments, DL-based attack detection mechanisms, and gaps. Next, it delves into the research methodology, describing the study design, chosen dataset, and research approach, while describing the proposed solution with the steps taken in developing and evaluating the Shallow-Deep ANN Hybrid Fusion models. Subsequently, the paper unveils the findings and results from experiments, presenting the hallow-Deep Hybrid Fusion model performance, providing valuable insights, while outlining potential future research directions, followed by the conclusion, summarizing the entire research work.

1. **Literature Review**

Edge computing architecture is a new paradigm offering a more efficient alternative to conventional cloud computing by addressing response bottlenecks and network delays in time-sensitive applications. It also helps preserve privacy by processing data closer to the source. Edge servers help reduce latency allowing for the seamless integration of IoT devices into crucial real-time applications in smart cities, industrial systems, healthcare, agriculture, and autonomous vehicles. They continuously generate, process, and transmit large amounts of sensitive data for intelligent decision-making. (Satyanarayanan, 2017). Edge servers help optimize traffic flows in smart cities and enhance public safety enabling real-time surveillance. They empower industries by predicting equipment failures and assisting in ensuring product quality. In agriculture, edge servers facilitate precision farming by monitoring crop conditions and optimizing resource usage. Moreover, edge servers enable continuous monitoring of patient health, among other enrichments (Qiu et al., 2020).

Security Challenges in Edge Networks

Edge networks employ diverse platforms and devices, which inherently encompass vulnerabilities and risks. Various communication protocols facilitate seamless interaction between IoT devices, edge application servers, cloud servers, service providers, and users. They are built on the TCP/UDP stack having several security challenges, such as the absence of default encryption, weak authentication, susceptibility to spoofing, and potential configuration errors due to the lack of universal security standards. These vulnerabilities in edge devices and protocols can render edge application servers susceptible to network attacks, requiring effective attack detection mechanisms. Therefore, robust security measures are essential to instill consumer confidence and foster growth and innovation in edge computing (Meegammana & Fernando, 2023). Conventional security mechanisms like firewalls, antivirus software, and IDS/IPS fall short of ensuring the security of edge application servers. This is primarily due to several key factors: the large attack surface, device heterogeneity, and increased physical exposure. Edge application servers are distributed across numerous locations, making attack monitoring difficult. Deploying in less secure environments makes them vulnerable to tampering. Moreover, the wide variety of hardware configurations, applications, and protocols supported by edge devices complicates the implementation of uniform security measures (Jin et al., 2022).

Neural Network Solutions

A neural network consists of interconnected neurons arranged into input, hidden, and output layers. Neurons receive inputs with associated weights, perform transformations on these inputs, and pass the results to neurons in the subsequent layer via forward propagation. The output layer generates predictions based on the processed information. The architecture of NN models, determined by the number of hidden layers and neurons, significantly influences their ability to learn complex patterns, resource requirements, and inference speed. The fundamental building block of a neural network is the neutron. It receives multiple inputs and associated weights, calculates the weighted sums of the inputs, adds bias, then applies an activation function to ensure nonlinearity, and subsequently generates the output to the neurons in the next layer as shown in Figure 2.0 (Aggarwal, 2018).

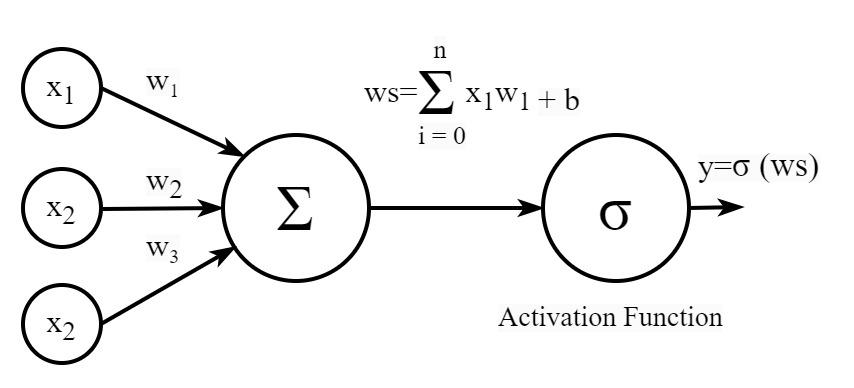


Fig. 2 Architecture of a Neuron

Neural Network techniques have been effectively adopted to detect network attacks on edge application servers. The capability of NN models to analyze network traffic patterns and anomalies enables the identification of suspicious activities and malicious threats in real-time, thereby enhancing the security posture of edge applications. Moreover, NN solutions can adapt and evolve learning from new attack patterns and data to offer proactive defense mechanisms against emerging threats (Moustafa et al., 2022).

Despite numerous advantages for network security, NNs also pose a variety of challenges. NN models demand large amounts of high-quality data to train effectively, and the acquisition of labeled data is time-consuming and expensive. Training neural network models often require significant computational resources and time. NNs are susceptible to overfitting and underfitting impacting generalizability to unseen data. Additionally, NN models often operate as black boxes, posing challenges in interpreting their decisions and outcomes (Aggarwal et al., 2023). Caviglione et al. (2023) conducted a survey highlighting NN model vulnerabilities to adversarial attacks, such as data poisoning, which result in incorrect predictions and emphasize the necessity for robust and resilient NN models. The lack of transparency in NN models can hinder adoption while raising ethical concerns related to privacy, bias, and fairness. Ensuring representative data, accountability, and transparency is imperative in the NN approaches (Sutaria, 2022).

Hybrid Models in Federated Learning

Federated learning (FL) aims to enable privacy by design, by reducing the risk of data breaches in the cloud servers. It takes a central NN model to local data to train it on a network of devices, and then only the models, not the data, are sent to the server. The server aggregates the received models into a global model, which is then sent back to the devices. FL process enhances model performance through shared and repetitive learning until convergence while maintaining user privacy (Abreha, Hayajneh, and Serhani, 2022). Nevertheless, FL also faces challenges of communications bottlenecks and potential adversarial attacks at local edge devices. A Shallow-Deep decision-level fusion model, combining the strengths of both models, aims to increase the performance, efficiency, and resilience of local models.

Similar Work

Sulieman et al. (2022) compare edge computing with cloud computing, focusing on use cases and architecture, while examining optimal server placement, cloud-edge network designs, and potential vulnerabilities. The survey presents a wide range of use cases based on IoT and 5G networks, encompassing latency-sensitive and bandwidth-constrained applications. However, the study overlooks the analysis of challenges and issues concerning edge deployment, heterogeneity, resource constraints in devices, and security including data privacy, as well as distributed management and monitoring aspects.

Zeyu et al. (2020) explore the emergence of the Internet of Everything, autonomous driving, immersive virtual and augmented reality games, and IoT enabled by edge computing technologies empowered by 5G networks. They emphasize the importance of security in the edge landscape, particularly in key management, privacy protection, attack mitigation, and anomaly detection. The gap in the above research lies in the absence of a solution-focused approach to propose specific strategies, methodologies, or technologies to effectively mitigate these challenges, leaving room for further research.

Alwarafy et al. (2020) offer a comprehensive examination of the security and privacy challenges within edge computing-assisted IoT. They highlight the drawbacks of centralized cloud computing while recognizing the heightened risks to data security and privacy in edge computing. The paper delves into security and privacy definitions, attack classifications, and potential solutions. However, including detailed case studies and real-world examples can enhance comprehension of security and privacy aspects in edge computing.

Kim and Lee (2022) analyze attack vectors and potential risks from malware attacks targeting industrial edge servers controlling IoT devices. They warn of the potential of physical damage to infrastructure. They developed a solution employing a Convolutional Neural Network (CNN) with the Malimg dataset for monitoring and analyzing edge server traffic, allowing for real-time malware detection. The study achieved higher classification accuracy. However, a weakness in their research lies in the class imbalance within the Malimg dataset, which can predominantly favor the majority class.

Jullian et al. (2023) implemented a Distributed Deep Learning (DDL) model utilizing Feed Forward Neural Network and Long Short-Term Memory (LSTM), employing NSL-KDD and BoT-IoT datasets for detecting attacks on edge servers. The research achieved a remarkable accuracy of 99.95%. However, the BoT-IoT dataset focuses on IoT botnet detection, while the SL-KDD dataset targets intrusion detection with network traffic attributes. Combining them as a single dataset for DL requires extensive preprocessing due to structural differences. The data normalization and class imbalances also pose further challenges. Therefore, the study lacks clarity on how the datasets were merged, leaving a gap in understanding the methodology. Additionally, the study does not detail how the FFN and LSTM detection results were combined, where a hybrid approach might have been more appropriate, but this isn't explained.

Wu, Wei, and Feng (2020) conducted a survey on recent research regarding DL for intrusion detection in computer networks and hosts. They explored key concepts, techniques, and advantages of various DL architectures, including supervised and unsupervised methods such as RNN, CNN, DBN, GAN, Autoencoders, and hybrid approaches. The review primarily focuses on important technical aspects However, there is a notable absence of a review of solutions among the reviewed works, which would enhance the practical effectiveness of these techniques and add significant value to the survey.

Zhang et al. (2022) investigated the application of Bidirectional Long Short-Term Memory (BiLSTM) DL models for detecting DDoS attack patterns in edge network traffic. They observed that BiLSTM outperformed RNN and LSTM models in this task. However, the rationale behind choosing BiLSTM, commonly employed in natural language processing tasks, was not provided in the study. While BiLSTMs can capture patterns from both directions of the input sequence, the study should acknowledge the added complexity and longer training time compared to RNN and LSTM models. Therefore, presenting additional performance metrics such as F1-score, CPU and memory usage, including training and prediction times would offer a more comprehensive comparison, especially across different attack scenarios.

Hybrid deep learning leverages the strengths of NN models by integrating them into a single model for improving overall performance and achieving better predictions. Ganaie et al. (2022) explore various ensembling methods, including bagging, boosting, stacking, decision fusion, and consensus employed in Hybrid NN applications. Abimannan et al. (2023) present findings from a survey, and discuss challenges, interpretability, adversarial security, and adaptability of hybrid models across diverse applications. However, neither study offers concrete measurements of the presented methods' outcomes, nor do they provide an analysis of pivotal techniques such as concatenation, weighted average, multiplication, subtraction, and other potent methods for constructing hybrid models.

Liu et al. (2020) conducted a federated learning (FL) experiment using a Convolutional Neural Network-Long Short Term Memory (CNN-LSTM) hybrid model for Anomaly Detection in IoT. They employed MNIST and CIFAR-10 datasets for edge anomaly detection among edge devices. The system involved multiple IoT devices that trained the shared model using local traffic data and then shared model parameters with the cloud server. The cloud server aggregated the received parameters to generate a new model and transmit it back to the IoT devices at the edge. The system aimed to preserve privacy and minimize communication overheads. However, MNIST and CIFAR-10 datasets, primarily designed for image classification tasks, may not fully represent the complexity and diversity of edge network traffic data. Augmenting these datasets with network traffic features could introduce biases, limiting the model's ability to generalize to real-world scenarios. CNN-Hybrid models are computationally expensive for Federated Learning, hence prone to creating bottlenecks when deployed on resource-constrained edge devices. Furthermore, a large number of edge devices will challenge communication costs. Although the model assumes all members to be legitimate, a malicious node can poison the central model with adversarial data. Hence, strategies for identifying malicious nodes, secure model updating, and transmission should have been considered.

Ullah et al. (2022) employed a hybrid Deep learning (DL) model based on Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU), utilizing a boosting approach with a car-hacking dataset for network attack detection in autonomous vehicles. The study achieved an accuracy of 99.9% in detecting Distributed Denial of Service (DDoS) attacks. However, the study lacks the provision of training and testing metrics for the hybrid model, which could aid in evaluating its performance.

Furthermore, Qazi et al. (2023) utilized a Hybrid Deep Learning (HDL) network consisting of a Convolutional Neural Network (CNN) and Long Short-Term Memory (LSTM). They combined an RNN with a CNN using the CICIDS-2018 dataset and achieved higher accuracies. Most research testing out different DL models to create Hybrid models has not tried Shallow-Deep approaches of the same model and using concatenation, weighted average, maximum, minimum, and multiplication methods experimented in this study.

Javeed et al. (2021) discuss the advantages of Hybrid NN models, which offer a flexible and scalable architecture for addressing some problems faced by single NN models employed in Software Driven Networks) based IoT environments. Hybrid models leverage the strengths of multiple individual models by combining them into a single, integrated system to enhance overall performance. These combined models can effectively tackle a wide range of challenges by utilizing the unique capabilities of each constituent model. Hybrid models facilitate transfer learning by using pre-trained components for specific tasks and adapting them for new tasks. They can also handle diverse types of data by integrating multiple specialized models. However, a gap in the research is the lack of consideration of the ultimate choice between a Hybrid and a single NN model, which depends on the complexity of the task, the availability of data, computational resources, and the desired level of interpretability. In some cases, a single NN model may be suitable, while in others, a hybrid approach may be required to achieve optimal performance. Therefore, the utilization of Cu-GRU LSTM and Cu-DNN LSTM hybrid models in low-resourced IoT devices introduces added complexity to low-resourced IoT devices. Hence, it requires a thorough assessment of resource utilization such as CPU and memory usage, power consumption, and performance metrics like latency and throughput to validate their efficiency, practicality, and suitability for real-world applications.

Khan et al. (2023) combined RNN and GRU algorithms in a hybrid NN model for attack classification using the ToN-IoT dataset. They found that the Adam optimizer performed better than Adamax. The hybrid model achieved 98% accuracy in the application layer and 99% accuracy in the network layer, outperforming other NN methods explored. However, while the RNN-GRU hybrid offers improved memory and temporal capabilities, it also increases model complexity, requiring careful hyperparameter tuning. Training stability may be compromised due to different learning rates between RNNs and GRUs. Additionally, the model may be computationally intensive for low-resourced edge devices and potentially prone to overfitting.

NN model fusion combines predictions from multiple NN models into a single prediction to achieve better performance by balancing the biases and errors inherent in individual models. Li et al. (2023) conducted a literature survey and discussed the challenges of NN model fusion, such as computational complexity, deciding on model selection, ensuring model diversity, limitations of data availability, risk of overfitting, increased latency, and implementation complexities. Their review explores parallel combinations of Shallow and Deep NN models comprehensively and presents novel approaches. Feature-level fusion extracts features from both Shallow and Deep models to use information from both models and feed them into a unified classifier for prediction. Decision-level fusion is a parallel process that feeds separate inputs into Shallow and Deep models and combines their predictions using different fusion functions like concatenation, maximum, minimum, and subtraction. Progressive stacking is a sequential process that utilizes the predictions of a Shallow model as extra features to train a Deep model, helping the Deep model learn from the errors made by the Shallow model to improve its performance. Hierarchical fusion builds a model hierarchy of models, integrating information from both models and then trains the Deep model using the outputs of the Shallow model. The selective ensemble combines predictions of Shallow and Deep models selectively using the difficulty of inputs or confidence level of each model. A gap in the study is the lack of practical comparison of hybrid methods in terms of implications on model complexity, inference time, and resource utilization in real-world scenarios, particularly edge environments.

Sedjelmaci et al. (2022) highlight a major weakness in explored NN solutions as the high false detection rate, particularly against zero-day threats. They propose hybrid NN approaches to minimize the false detection rate and defend edge networks from known and unknown attacks. They use federated learning-based hybrid models implemented at each edge server to make prediction decisions through cooperation between distributed security engines, employing a trusted security game approach. While this study offers a novel approach, a gap exists in possible model poisoning by malicious nodes. Hence, strategies for protecting local nodes are imperative against adversarial attacks.

Research Gaps

This review on network attack detection in edge environments identifies key insights and research gaps. While Satyanarayanan (2017) and Qui (2020) introduce edge computing architecture and use cases, they overlook security implications and challenges in edge network deployment. Sulieman et al. (2022) provide valuable insight into the advantages of edge computing and deployment challenges, comparing it with cloud computing. Meegammana and Fernando (2023) emphasize the need for robust security measures in edge networks and highlight the potential of deep learning for mitigation. Moustafa et al. (2022) propose NN approaches to address security challenges in edge devices and protocols, while Aggarwal et al. (2023) discuss NN challenges in network attack detection. Caviglione et al. (2023) highlight the risks of adversarial attacks on NN models, and Sutaria (2022) raises ethical concerns regarding NN model development. Zeyu et al. (2020) stress the importance of security in autonomous vehicles enabled by edge technologies, indicating the need for further research. Alwarafy et al. (2020) recognize heightened risks to data security and privacy in edge computing. Kim and Lee (2022) propose a CNN-based solution for malware attacks on edge servers, overlooking class imbalance. Jullian et al. (2023) achieve high accuracy in attack detection using federated learning but lack clarity on merging multiple datasets. Wu et al. (2020) review deep learning for intrusion detection, missing practical insights on deployment challenges in edge environments. Zhang (2022) explores important hybrid techniques but lacks their practical comparison in edge environments. Ganaie et al. (2022) explore various ensembling hybrid methods, while Abimannan et al. (2023) discuss the challenges of hybrid models but lack practical exploitation in edge networks.

Liu et al. (2020) implement federated learning for attack detection in the edge network but overlook scalability and security concerns. Ullah et al. (2022) achieve impressive results using LSTM and GRU hybrid NN for network attack detection but lack key performance metrics to assess its deployment potential in edge environments. Javeed et al. (2024) discuss the advantages of Hybrid NN models but lack a strategy for choosing them over single models. Khan et al. (2023) achieve high accuracy using the RNN-GRU hybrid but may face computational challenges in low-resource edge environments. Li et al. (2023) discuss various NN model fusion methods theoretically providing novel insights for practical exploitation. Sedjelmaci et al. (2022) propose federated hybrid neural network approaches but overlook potential malicious nodes and model poisoning risks. Overall, more comprehensive studies addressing model development, interpretability, scalability, security, and real-world applicability are required. Previous studies focused on various hybrid NN models but not Shallow-Deep hybrid NN models employing parallel fusion methods, which is a novel approach in this study. This research aims to bridge these gaps by developing optimized Shallow-Deep hybrid NN models employing parallel fusion methods for network attack detection on edge application servers.

1. **Methodology**

Research Process

The study followed an ML pipeline shown in Figure 3.1, comprising a predefined series of interconnected steps aimed at producing a high-quality final product (Hapke & Nelson, 2020). This approach builds upon the prior work of Meegammana and Fernando, (2024) to develop multiple Shallow-Deep Hybrid Fusion models to enhance security on edge application servers against network attacks.

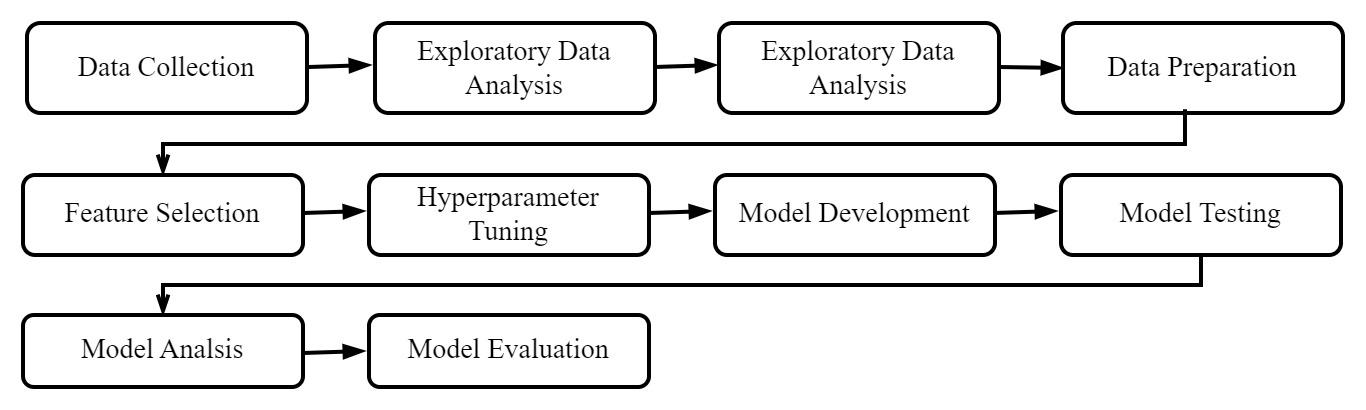


Fig. 3 Research Process

Data Collection

The dataset plays a pivotal role in laying the foundation of NN research for determining the model performance. In this study, a subset of the widely known UNSW-NB15 dataset was utilized, comprising 257,673 instances of network traffic data with 45 features, covering 9 attack classes and normal data. The target feature, labeled as "attack" and "benign" classes, facilitates supervised learning tasks (David, 2019). This diverse and representative dataset has enabled the development of robust models capable of generalizing well to real-world scenarios and also ensured fairness and ethics to explore novel techniques. This subset was particularly selected to mitigate computational intensity and training time while remaining relevant to the study. During exploratory analysis, a class imbalance was identified in the target variable "label", potentially biasing the model training toward the majority "benign" class. The study employed Random undersampling to address this issue by balancing the "attack" and "benign" classes. This resulted in reducing the dataset to 186,000 instances and enabled ensuring effective learning from both classes, to enhance model performance.

Characteristics of the Dataset  
The UNSW-NB15 dataset consists of 9 attack categories and begin instances described in Table I.

Table I

Attack Categories in the UNSW-NB15 Dataset

|  | **Attack** | **Description** |
| --- | --- | --- |
| 1 | Fuzzers | Pose a threat to IoT applications by injecting random data, potentially leading to processing failures or causing application servers to crash. |
| 2 | Analysis | Conduct active reconnaissance, port scans, and vulnerability assessments to identify potential vulnerabilities for exploitation. |
| 3 | Backdoor | Install malicious software to establish unauthorized remote access to application servers. |
| 4 | Denial of Service (DoS) | Exploit cryptographic weaknesses to target poorly encrypted data blocks, streams, or messages. |
| 5 | Generic | Leverage cryptographic principles to target weakly encrypted data blocks, streams, or messages. |
| 6 | Reconnaissance | Passively gather preliminary information about target hosts and utilize publicly accessible data sources for intelligence gathering. |
| 7 | Shellcode | Inject payloads to compromise target systems, granting attackers remote access to a command shell within application servers. |
| 8 | Worms | Spread malware via networks, converting infected IoT devices into botnet zombies for executing distributed attacks. |
| 9 | Exploits | Exploit known vulnerabilities in operating systems or applications to gain unauthorized access for malicious activities. |

Figure 2.0 illustrates the distribution of 9 types of attacks within the dataset after class balancing.

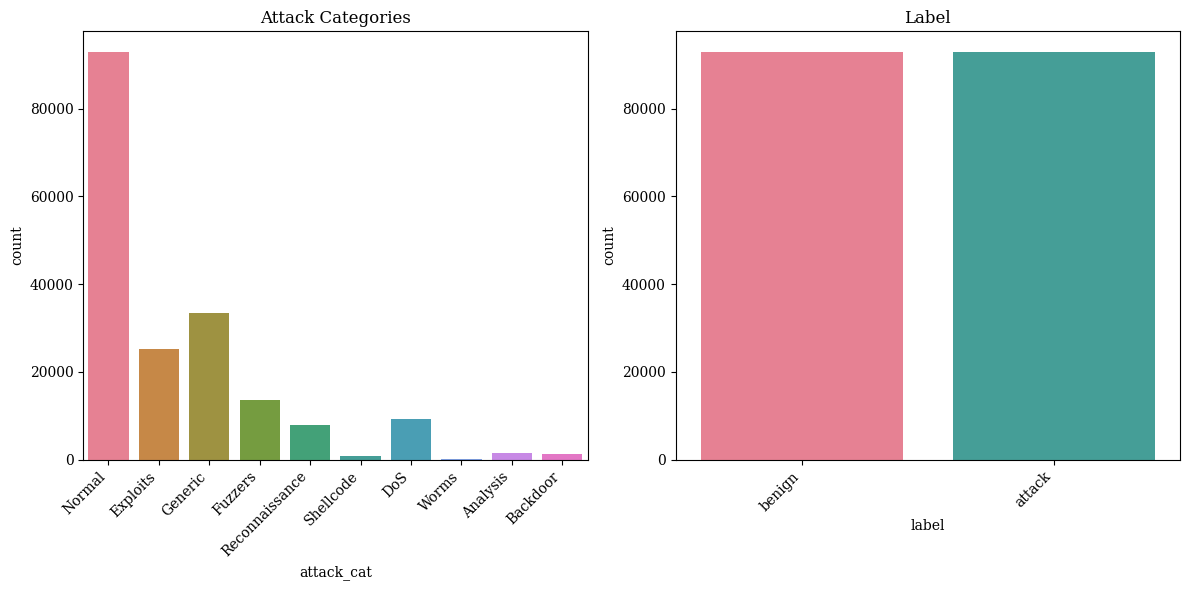


Fig. 4 Distribution of Instances

The attack categories in the dataset include traffic samples of reconnaissance, DoS attacks, fuzzes, shellcode, backdoors, weakly encrypted data, injected payloads, malware, known vulnerabilities, including network analysis, and generic exploits. They represent diverse network threats targeting unauthorized access, data interception, privilege escalation, data theft, or system disruption, thereby posing substantial risks to application servers within edge environments.

Data Splitting

The study followed standard machine learning practices to partition the dataset into training, testing, and validation sets. Approximately 90% of the data was allocated for training, while 5% each was allocated for testing and validation. This allocation ensures that the model can effectively learn the underlying patterns and relationships within the dataset. Allocating a larger portion of data for training, enhanced the predictive performance of models enabling learning from ample numbers of examples. The reserved 5% for testing allowed for the evaluation of the model's performance on an independent dataset not encountered during training, as well as provided an accurate estimate of its generalization to new examples. Similarly, the 5% allocated for validation helped in monitoring the model's performance during training, optimizing hyperparameters, and preventing overfitting. Overall, this data-splitting strategy ensured sufficient data availability for model training, robust evaluation, and effective performance monitoring while maintaining a balance between these aspects.

Feature Selection

This study derived two datasets comprising 20 significant features (Meegammana and Fernando, 2023), and 40 numeric features from the preprocessed UNSW-NB15 dataset. This approach aimed to investigate the effects of low and high-dimensional data on model performance in different environments. The dataset containing 20 features may be suitable for deployment in low-resource edge environments, helping to reduce model complexity. Likewise, the dataset containing 40 features may be better suited for attack detection on high-end edge servers.

Data Normalization

The study employed the Min-Max Scaling normalization technique to bring all features to a common scale by transforming their values, typically between 0 and 1. This process ensured that all features contributed equally during model training and evaluation. Figure 3.0 illustrates the effect of normalization on a dataset. It is also beneficial when dealing with features that have different ranges or units (Goodfellow et al., 2016).

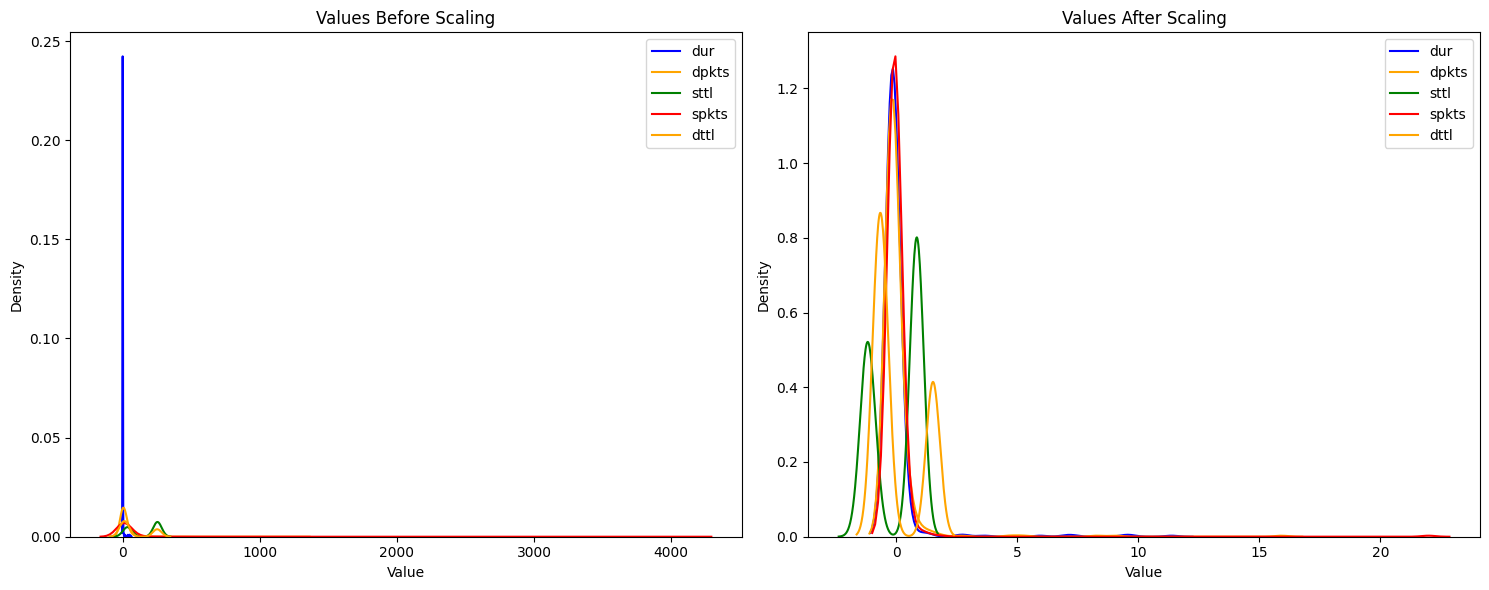


Fig.5 Data values before and after normalization

As shown in Figure 4.0, the Min-Max Scaling technique helped to improve model convergence by scaling features to a fixed range, and it preserved the distribution of the original data to maintain interpretability. It also retained the relative positions of outliers or extreme values.

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

x\_scaled = (x - min(x)) / (max(x) - min(x))

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.

Shallow and Deep Model Architectures

The study utilized Shallow and Deep NN models to construct Hybrid Fusion models. The Shallow NN model, simpler in design, consists of a single hidden layer with 512 neurons. In contrast, the Deep NN model exhibited a more intricate architecture, consisting of 7 hidden layers with 256, 128, 64, 32, 16, 8, and 4 neurons respectively. Figure 5 illustrates the architectures of the Shallow and Deep models used in the study.

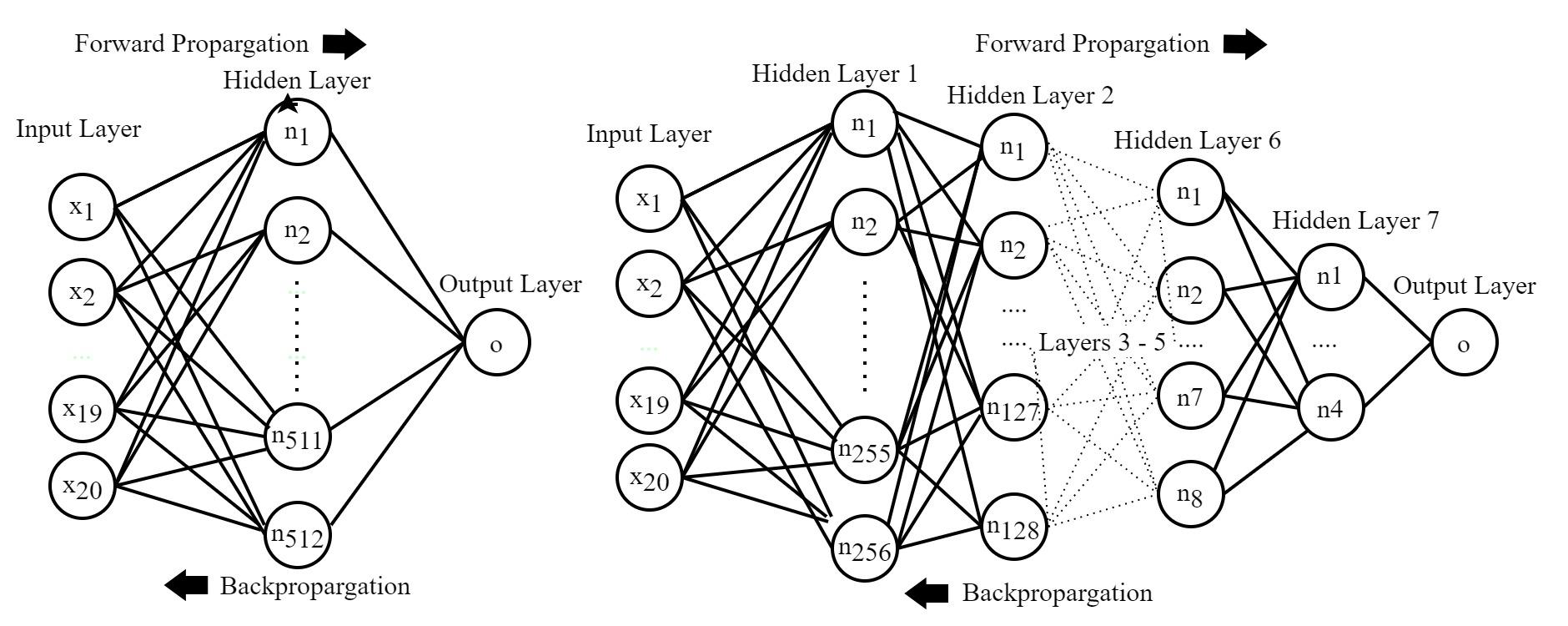


Fig. 6 Shallow and Deep ANN Model Designs (Meegammana and Fernando, 2024)

The Shallow NN model was designed to capture primary patterns and relationships within the data while maintaining computational efficiency. Its architecture strikes a balance between capturing complexity and simplicity by employing only one hidden layer. This simplicity facilitates easier interpretation of the model's behavior. In contrast, the Deep NN model, comprising 7 layers, allows for the extraction of hierarchical features, and capture of intricate patterns and nuances present in the data through multiple layers. The increased depth of the Deep model also introduces challenges such as vanishing gradients and overfitting. These challenges are addressed in the study through regularization techniques L1, L2, and early stopping with hyperparameter optimization (Meegammana & Fernando, 2024).

Each hidden layer in models uses an activation function to enable crucial non-linearity, which allows a model to learn complex patterns and relationships in the data. This study employed Rectified Linear Unit (ReLU), tanh, and Leaky ReLU activation functions, determined through hyperparameter tuning (Aggarwal, 2018). The Sigmoid function was only used for the output layer. Their behaviors are illustrated in Figure 6.0 and the formulas are presented in Table 2.0.

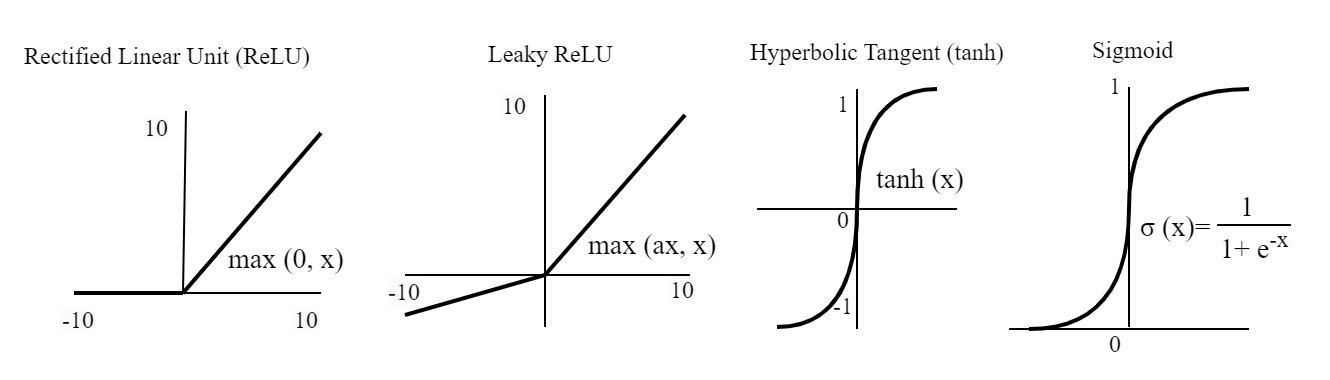


Fig. 7 Activation Functions Used in the Study.

Table II

Activation function formulas

| **Activation Function** | **Formula** | **Output Range** |
| --- | --- | --- |
| Rectified Linear Unit (ReLU) | f(x) = max(0, x) | [0, ∞ ] |
| Leaky ReLU | f(x)=max(ax, x) | [0, ∞] for x>=0 and [∞, -∞] for x < 0 |
| Hyperbolic Tangent (tanh) | f(x) =(ex-e-x )/(ex+e-x) | [-1, 1] |
| Sigmoid | f(x) =1/(1+e-x) | [0, 1] |

During hybrid model training, the study employed the Adam optimization algorithm to adjust its weights and minimize error. It quantifies the disparity between the model's predictions and the actual results, subsequently updating the network's weights based on the gradients of the loss function (Goodfellow et al., 2016).

The formulas for the Adam optimization algorithm are given below.

m\_t = β\_1 \* m\_{t-1} + (1 - β\_1) \* g\_t

v\_t = β\_2 \* v\_{t-1} + (1 - β\_2) \* g\_t^2

hat{m}\_t = m\_t / (1 - β\_1^t)

hat{v}t = v\_t / (1 - β\_2^t)

θ{t+1} = θ\_t - (α \* hat{m}\_t) / (√(hat{v}\_t) + ε)

where:

m\_t and v\_t are estimates of the first moment (the mean) and the second moment (the uncentered variance) of the gradients respectively,

β\_1 and β\_2 are the exponential decay rates for the moment estimates (typically set to 0.9 and 0.999 respectively),

g\_t is the gradient of the loss function at time step t,

α is the learning rate,

θ\_t are the parameters of the neural network at time step t,

ε is a small constant added to the denominator for numerical stability.

This study utilized the Binary Cross-Entropy Loss (Log Loss) function. It helps in training the model by providing a measure of how well the model's predictions match the actual labels and guiding the optimization process to improve the model's performance.

The Binary Cross-Entropy Loss formula is given below.

L=1/n ∑ ['y log(ŷ) + (1-yi) log(1-ŷi)

Where:

n is the number of samples.

𝑦𝑖 is the actual binary label of the i-th sample (0 or 1).

ŷi is the predicted probability of the i-th sample in class 1 (0 to 1).

The logarithmic terms in the formula ensure increased loss significantly when there is a large difference between the predicted probability and the actual label.

Hyperparameters Tuning

The study used the optimized hyperparameters to build the Shallow and Deep models given by Meegammana and Fernando (2024). They ensured that the hybrid models constructed using these models benefit from optimized Shallow and Deep model configurations. They employed Keras random search for hyperparameter tuning, while keeping the input features, number of hidden layers and neurons, and output layer constant. Obtained hyperparameters involving activation functions, dropout rates, regularization settings, weight initializer, optimizer, learning rate, and batch size. The hyperparameters used to construct Shallow and Deep models in the study are shown in Table 3.0.

Table III

Optimized Architectures of the Models after Hyperparameter Tuning

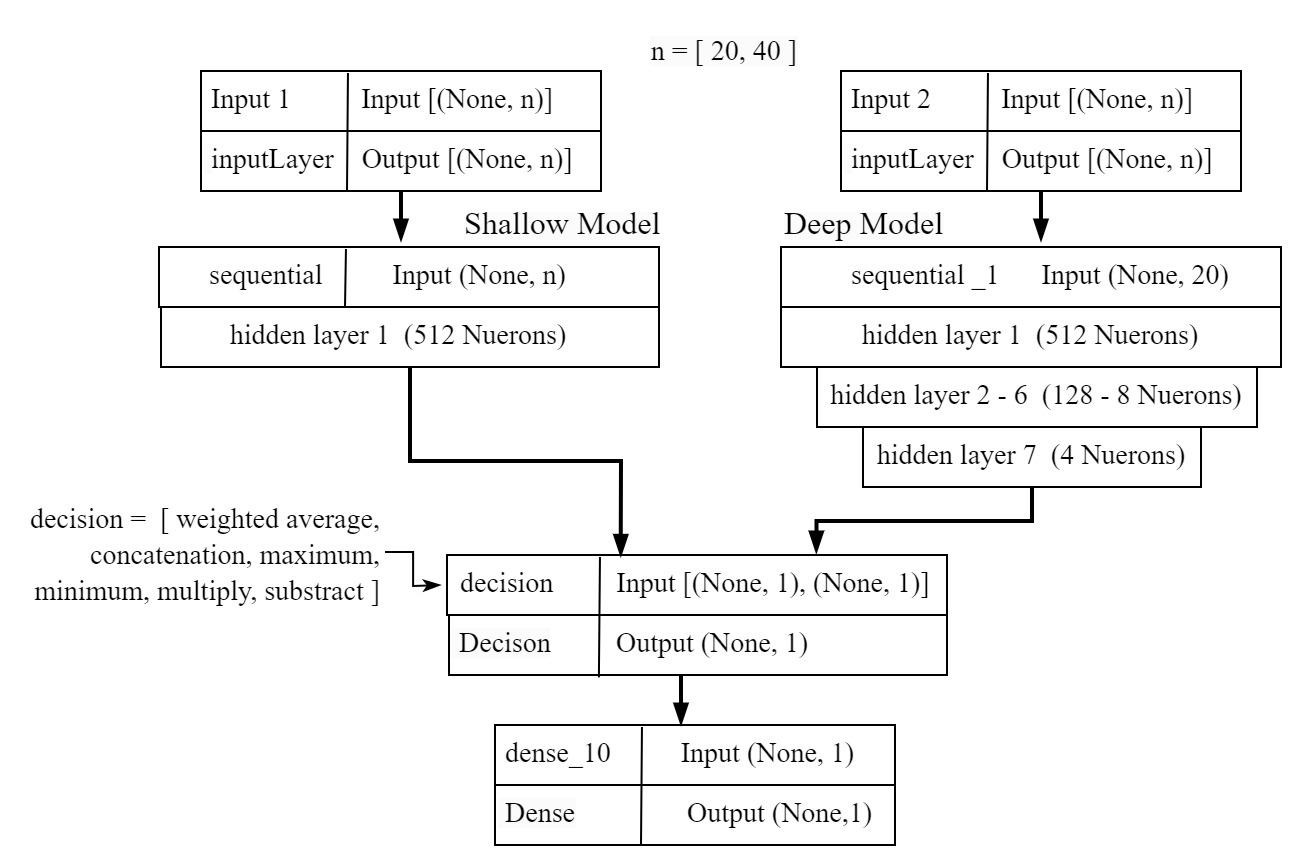
| **Configuration** | **Shallow20** | **Shallow40** | **Deep20** | **Deep40** |
| --- | --- | --- | --- | --- |
| Model Parameters | 11265 | 49313 | 21008 | 54433 |
| Data Inputs | 20 | 40 | 20 | 40 |
| Hidden layers | 1 | 1 | 7 | 7 |
| Neurons per layer | 512 | 512 | 256, 128, 64, 32, 16, 8, 4 | 256, 128, 64, 32, 16, 8, 4 |
| activation\_layer1 | Relu | Tanh | tanh | ReLU |
| activation\_layer2 | - | - | tanh | ReLU |
| activation\_layer3 | - | - | ReLU | leaky\_relu |
| activation\_layer4 | - | - | ReLU | ReLU |
| activation\_layer4 | - | - | 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 |

Experimental Models

The study employed configurations of four primary models: Shallow20, Deep20, Shallow40, and Deep40, developed by Meegammana & Fernando (2024), formed the foundation for constructing 12 diverse Hybrid Shallow-Deep Fusion NN models. Subsequently, these hybrid models underwent training employing 20-feature and 40-feature datasets for evaluating network attack detection on edge servers.

Hybrid Model Creation

The study followed the parallel ensemble learning approach to build the 12 Hybrid Fusion NN models combining the independently trained Shallow and the Deep models applying decision fusion functions. This technique feeds the separate but similar input data through both the Shallow and Deep models in parallel. Then, combine their prediction using a Decision Fusion function (Choi, 2020), to produce the final prediction using the sigmoid output layer. Fusion functions explored in constructing hybrid models included weighted averaging, concatenation, maximum, minimum, multiple,y, and subtract techniques. This approach effectively addresses the shortcomings in the functionality of individual models to improve the overall robustness and prediction accuracy significantly. Figure 6.0 illustrates the development workflow and architecture of 12 Hybrid Fusion models. Finally, the output from Hybrid Fusion models was input to the sigmoid layer to obtain the prediction.

  
Fig. 8 Shallow-Deep Hybrid Fusion Model Architecture

Hybrid Fusion Model Inputs and Outputs

As illustrated in Figure 6, the n input features with shape (None, n) tensors, are fed into the Shallow and Deep Models in parallel, where n refers to the number of features in the dataset (20 or 40). The Shallow and Deep models process the input tensors, then output two predicted tensors with shape (None, 1). The decision fusion function taking these two tensors, having (None, 1) shape, processes and outputs a prediction tensor with shape (None, 1), containing the element-wise decision values (e.g., the maximum of the two inputs). The final prediction is made by a dense layer with 1 unit, applying a linear transformation to the input tensor with shape (None, 1). This transformation is defined by the weight matrix 'W' and bias 'b' associated with the layer, mathematically expressed as y = Wz + b. The output of this linear transformation is then passed through a sigmoid activation function (expressed as f(x) =1/(1+e-x) ) to produce the final prediction (Li et al., 2023).

Decision-level Fusion Functions and formulas are as follows:

Let x represent the input data.

Let Sp and Dp be predictions from the Shallow and Deep models.

Let x represent the combined prediction obtained by the fusion method.

Let h(z) represent the processing function.

Let ŷ be the final prediction using the sigmoid output layer.

Shallow Model Prediction: Sp = f\_S(x)

Deep Model Prediction: Dp = f\_D(x)

Fusion output: z = FusionFunction(Sp, Dp)

Here the FusionFunction can be replaced with either weighted averaging, concatenation, maximum, minimum, multiply, and subtract functions (Choi, 2020).

Final Prediction: ŷ = sigmoid(h(z))

The individual fusion function descriptions and formals are given below.

Let y represent the final fused output.

Weighted Averaging

This function combines the outputs of the models applying weighted sum, where the weights are predefined based on the complexity of each model. This study predefined weights of 0.15 and 0.85 for Shallow and Deep models respectively based on the number of hidden layers (1:7).

y= Z(w1.Sp + w2.Dp)

where,

w1 and w2 are pre-defined weights.

Concatenation

This function combines the outputs from multiple models by joining them end-to-end along a specified dimension.

y=Concatenate(Sp,Dp)​

Maximum

The function obtains the final output by taking the element-wise maximum of the outputs from the models.

y=maximum(Sp, Dp)

Minimum

The function obtains the final output by taking the element-wise minimum of the outputs from the models.

y=minimum(y1, y2)

Multiply

The function obtains the final output by taking the element-wise product of the outputs from the models.

y=(y1\*y2)

Subtract

The function obtains the final by taking the element-wise difference of the outputs from the models.

y=(y1-y2)

Final Hybrid Fusion models

The study used Shallow and Deep models, and 20-feature and 40-feature datasets, and applied 6 decision-level fusion functions to develop 12 Hybrid Fusion NN models as illustrated in Figure 7.0.

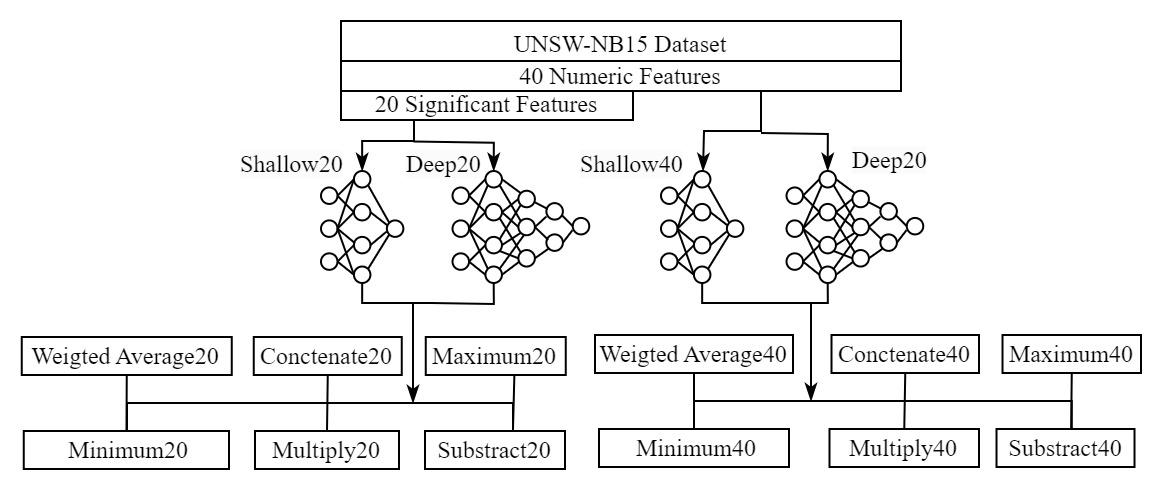


Fig. 9 Architectures of 12 Hybrid Fusion Models

Training the Models:

The study trained 12 Hybrid Fusion NN models employing 20-feature and 40-feature datasets. Their training performance was monitored in real-time using the validation dataset to prevent overfitting. Early stopping with a patience setting of 50 was implemented, to enable the training process to save the model at the epoch with the minimum validation loss. This strategy helps mitigate overfitting and underfitting effectively, and ensures effective hyperparameter utilization, while promoting generalization to achieve optimal performance on unseen data (Goodfellow et al., 2016 ).

Model Testing and Evaluation:

Each Hybrid Fusion NN model was constructed using their best configurations, trained and saved, and subsequently tested employing the dedicated test data unseen by the models during training. The performance metrics obtained from the training and testing process indicate the effectiveness of models in real-world scenarios (Chollet, 2018), finally guiding the study to select the most effective deployment model.

Performance Metrics

The study considered various performance metrics including validation loss, validation accuracy, test accuracy, precision, recall, and F1-score, along with model size, CPU, and memory usage obtained through training and testing of models. The validation accuracy and validation loss curves provided insights into the models' performance trends throughout the training process. These metrics were used collectively to assess model capabilities in generalization, pattern detection, and prediction.

Confusion Matrix

The confusion matrix shown in Figure 8 offered a comprehensive overview of predicted versus actual values, summarizing the classification results for assessing the model's predictive performance. It included counts of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) predictions made on the dedicated test dataset, allowing computing key performance metrics and enabling a detailed evaluation of the model's accuracy (Chollet, 2018).

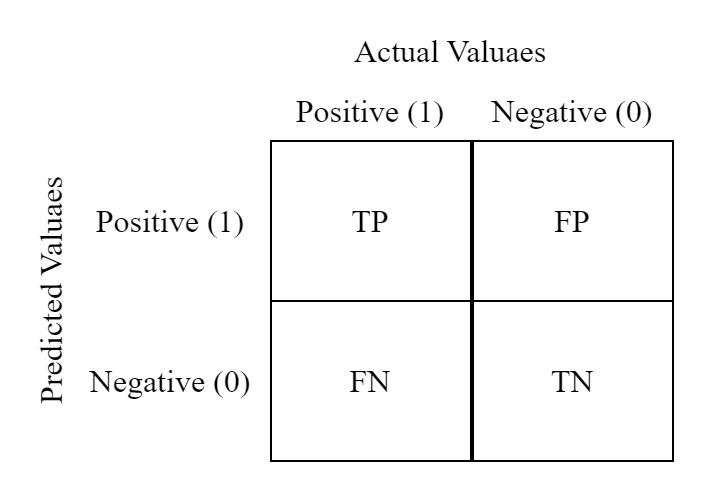


Fig. 9 The Confusion Matrix

The study computed the ROC Curve and AUC Score for each Hybrid Fusion model using test data to evaluate the model's performance on unseen instances by plotting the true positive rate (TPR) against the false positive rate (FPR) at various classification thresholds.

The study measured the time consumed for testing and predicting 9300 samples by each model, by running compiled model executables. This measure is important for assessing the real-time performance and scalability of the model. The study also obtained CPU and memory usage by each model using the resource monitor to evaluate the resource utilization of models. Ultimately, these metrics collectively helped to optimize model performance and make informed decisions about the choice of the best model for deployment considering the specific characteristics of application servers operating in specific edge environments.

Model Deployment and Monitoring:

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.

1. **Results and Discussion**

These results were obtained by testing Shallow-Deep Parallel Hybrid Fusion models. They were created by combining Shallow and Deep models and employing 20-feature and 40-feature datasets. The 12 models were named by concatenating fusion functions with several features used (i.e. Concat20, Concat40). The training and testing results of 6 models employing a 20-feature dataset are shown in Table IV.

Table IV

20-Feature Hybrid Model Performance

| Model | Shallow20 | Deep20 | WgtAv20 | Concat20 | Minimum20 | Maximum20 | Multiply20 | Subtract20 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| End epoch | 14 | 128 | 217 | 113 | 142 | 190 | 165 | 174 |
| Training time (M) | 7.28 | 29.54 | 74.23 | 40.89 | 60.09 | 48.71 | 48.69 | 71.89 |
| Validation Loss | 0.17 | 0.12 | 0.12 | 0.13 | 0.13 | 0.13 | 0.13 | 0.13 |
| Test Accuracy | 0.93 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 |
| Precision | 0.95 | 0.97 | 0.96 | 0.96 | 0.97 | 0.97 | 0.96 | 0.96 |
| Recall | 0.91 | 0.93 | 0.94 | 0.93 | 0.93 | 0.93 | 0.93 | 0.94 |
| F1 Score | 0.93 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 | 0.95 |
| ROC AUC Score | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 | 0.99 |
| Prediction Time | 0.52 | 0.62 | 0.71 | 0.86 | 0.71 | 0.71 | 0.69 | 0.99 |

Figure 14 shows the live monitoring results in the change of validation loss and test accuracy of 20-feature fusion models during training. Their training was halted when validation loss was no longer improving, to prevent potential overfitting.

|  |  |
| --- | --- |

Fig. 14 Validation Loss and Training Accuracy of 20-feature modes

The training and testing results of 6 models employing a 40-feature dataset are shown in Table 5.0.

Table V

40-feature hybrid model performance

| Model | WgtAv40 | Concat40 | Minimum40 | Maximum40 | Multiply40 | Subtract40 |
| --- | --- | --- | --- | --- | --- | --- |
| End epoch | 90 | 50 | 81 | 74 | 131 | 141 |
| Training time (M) | 28.86 | 18.91 | 24.88 | 38.26 | 57.81 | 70.18 |
| Validation Loss | 0.05 | 0.05 | 0.06 | 0.05 | 0.06 | 0.06 |
| Test Accuracy | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 |
| Precision | 0.98 | 0.98 | 0.99 | 0.98 | 0.99 | 0.98 |
| Recall | 0.98 | 0.98 | 0.98 | 0.98 | 0.97 | 0.98 |
| F1 Score | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 | 0.98 |
| ROC AUC Score | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 | 1.00 |
| Prediction Time | 0.82 | 0.71 | 0.63 | 0.73 | 1.29 | 1.17 |

Figure 15 shows the live monitoring results on the change of validation loss and test accuracy of 40-feature models during training. The training was halted when validation loss was no longer improving to prevent potential overfitting.

|  |  |
| --- | --- |

Fig. 15 Validation Loss and Training Accuracy of 40-feature modes

Training Time

The individual shallow model, employing 186,000 samples split into 90:5:5 training, testing, and validation sets, achieved the fastest training time, completing training in 7.28 minutes and converging in 14 epochs. In contrast, the deep model, despite its faster training consuming 29.54 minutes, required significantly more epochs (128) to converge. Hybrid fusion models exhibited longer training times as shown in Figure 16, and required a higher number of epochs to converge, while the Subtract40 model experienced the longest training consuming 70.18 minutes.

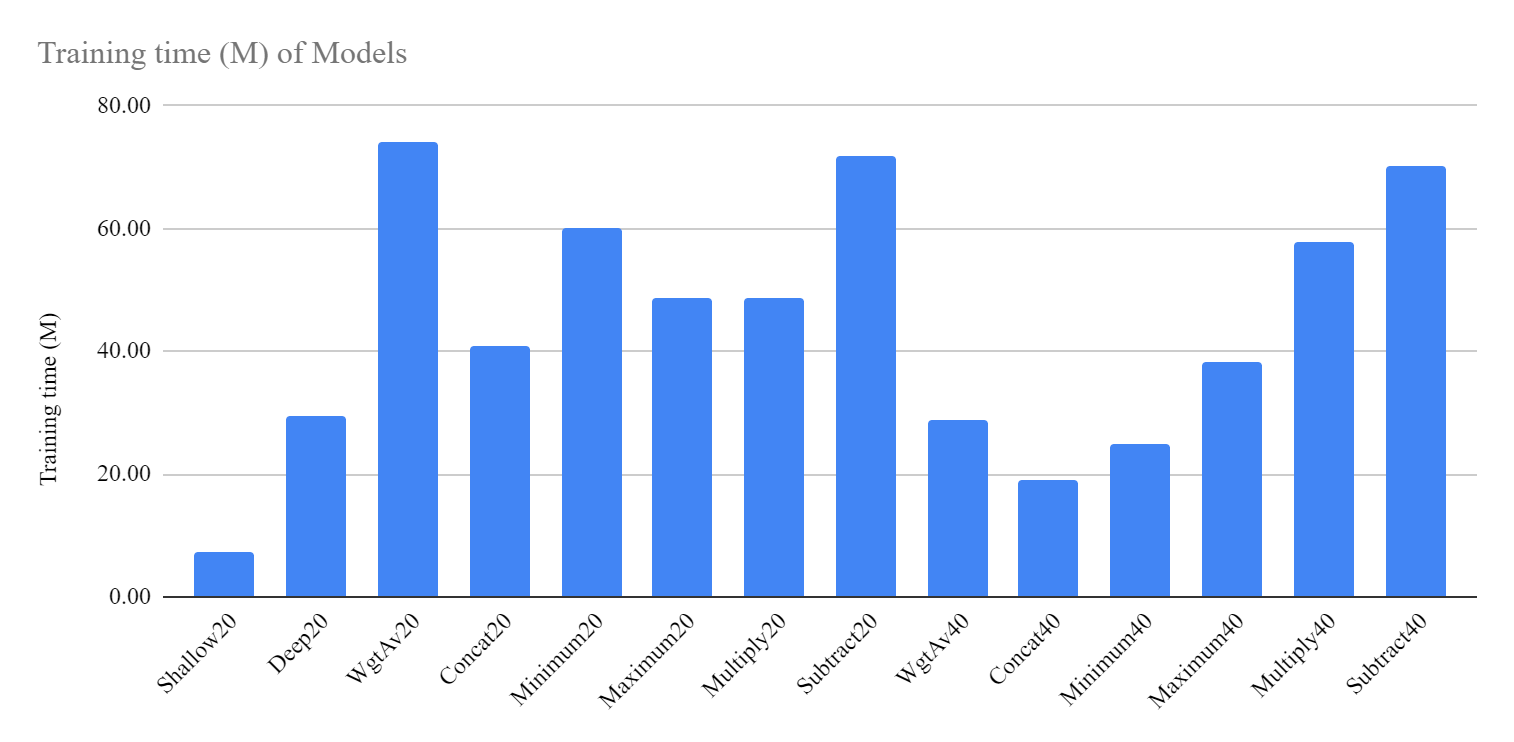


Fig. 16 Training time of Hybrid Fusion models

One-Time Learning and Federated Learning

The training time consumed by a model affects both one-time training and federated learning approaches. In a one-time learning approach, a model is trained once and then deployed in edge servers for inference. Longer training times can impede frequent model updates required to adapt to new types of network attacks. Federated learning distributes model training across multiple edge servers, and only model updates are aggregated on the central server (Abreha, Hayajneh, and Serhani, 2022). Longer training times with higher numbers of epochs, such as in the Substract40 model, can be particularly detrimental. These extended local training times can slow down the synchronization of updates across all devices, thereby delaying the overall convergence of the global model. This is particularly problematic for edge devices with limited computational capabilities, and the federated learning process can significantly degrade. Therefore, models with higher training times are not ideal for federated learning employing low-resource edge devices. For both one-time learning and federated learning, models like Shallow20, Maximum20, Concat40, Minimum40, and Maximum40 provide a good balance between training time and performance. These models ensure faster deployment and more efficient updates in federated learning. Although Maximum20 may offer slightly lower accuracy compared to models using 40 features, its faster convergence, moderate performance metrics, and training and prediction times help reduce synchronization delays. Hence they can improve the overall efficiency of the federated learning process.

Validation Loss

A higher validation loss can impact both federated and one-time learning causing poor model generalization, aggregatio,n and slowing convergence (Chollet, 2018). It also suggests potential overfitting, leading to poor performance when deployed to edge devices. All 40 feature Hybrid fusion models achieved significantly lower validation loss between 0.05 and 0.06. The validation loss recorded by 20 feature models is between 0.12 and 0.17 which is comparatively significant. Hence, Maximum40 and Concat40 hybrid models are more suitable for both federated and one-time learning when validation loss is considered.

Prediction Time

Prediction time is a critical factor in evaluating the performance of NN models used in edge servers requiring real-time processing and decision-making (Chollet, 2018), optimized resource utilization,n, and scalability. Figure 17 shows the prediction time in seconds to predict 9,300 samples by all models.

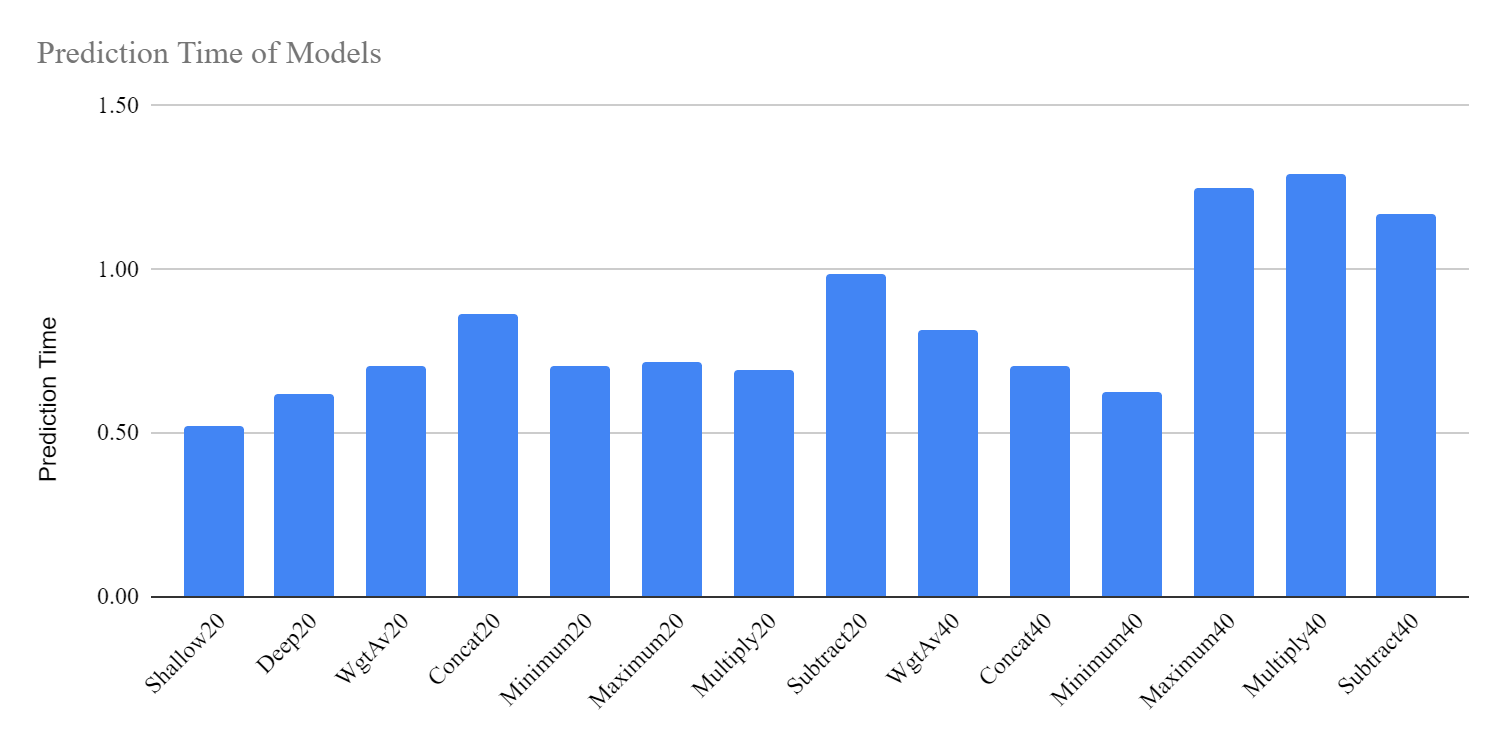


Fig. 17 Prediction time of Hybrid Fusion models

The shallow20 individual model recorded the lowest prediction time with 0.52 seconds for testing 9300 samples, while Concat40 and Minimum40 Hybrid models recorded 0.63 and 0.71 seconds for the same task. Although the difference seems negligible, when scaled edge environments such as autonomous vehicles require predicting vast numbers of traffic samples to detect an attack, then it becomes a significant response delay.

Test accuracyThe test accuracy is crucial as it measures the model's performance on unseen data, indicating its ability to make correct predictions in real-world scenarios (Chollet, 2018). A high test accuracy is important for deploying the model to edge servers, to ensure accurate predictions.

In this case, the Deep20 individual model achieved 0.95 accuracy, while the 40-feature Hybrid Fusion models generally achieved higher accuracy around 0.98, indicating their effectiveness in making correct predictions. However, test accuracy alone does not help in evaluating a model.

Precision

Precision indicates a model's ability to make accurate positive predictions and avoid false positives (Chollet, 2018). The 40-feature hybrid models achieved generally higher precision between 0.98 and 0.99 over other models.

Recall

Recall indicates a model's ability to avoid false negatives (Chollet, 2018). 40-feature Hybrid models generally achieved higher recall with 0.97 and 0.98 over 20-feature models.

F1 Score

The F1 score combines both precision and recall into a single value to give an overall measure of a model's accuracy in binary classification tasks (Chollet, 2018). It considers both false positives and false negatives which are incorrect predictions for calculating the F1 score. The 40-feature hybrid models generally achieved a higher F1 score of 0.98, while 20-feature models achieved 0.93 - 0.95, indicating the superiority of 40-feature models.

ROC AUC Score

The ROC AUC score evaluates the ability of a binary classification model to discriminate between positive and negative classes across different thresholds. All models demonstrated very high ROC AUC scores over 0.99, where 40-features hybrid models achieved nearly 1.0.

Radar Analysis  
Figure 18 presents a radar chart comprising the key performance metrics of 6 Hybrid Fusion models using 20 features. This analysis compares end epoch, training time, validation loss, testing time, testing accuracy, precision, recall, and F1 score obtained through training and testing of models. To obtain this radar chart, all results are normalized using min-max normalization, scaling each metric between 0 and 1.

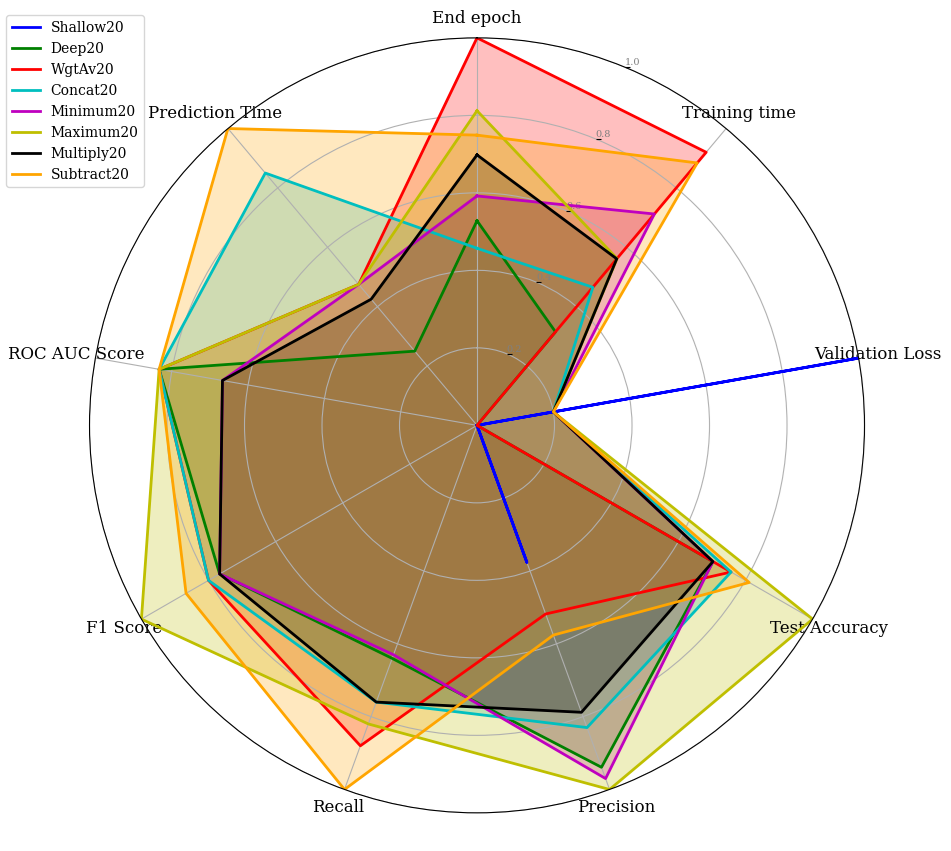


Fig. 18 Key Performance Metrics of 20-feature models

The radar comparison used a ranking strategy assigning ranks ranging from 1 to 6 to model metrics. Lower rankings were assigned for lower values indicating better performance in metrics such as end epoch, training time, validation loss, and prediction time. Conversely, lower rankings were assigned for higher values indicating better performance in metrics such as test accuracy, precision, recall, F1 score, and ROC AUC score.

To determine the best 20-feature model, the metrics of the models were ranked in ascending order, and the sum of these ranks was calculated. The overall ranks computed are as follows:

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

Based on these calculations, Maximum20 was the best-performing model, followed by Concat20 and Minimum20. Figure 19 presents a radar chart comparing the key performance metrics of 6 models using 40 features involving the same performance data.

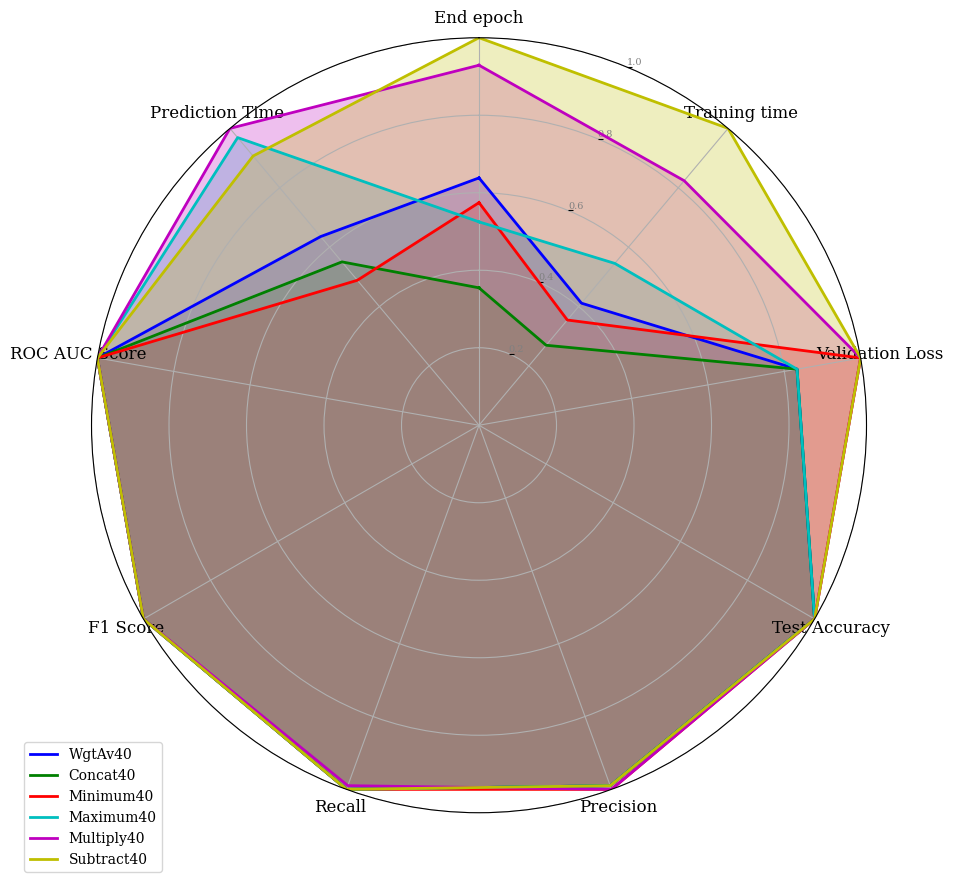


Fig. 19 Key Performance Metrics of 40-feature models

Using a similar ranking approach, overall ranks were calculated as follows:

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

Based on these calculations, Concat40 was the best-performing model, followed by WgtAve40 and Minimum40. Among 20-feature and 40-feature Hybrid Fusion models Concat40 is the best performing model.

Resource Usage

The study evaluated 9 selected high and low-performing models in terms of model size, including CPU and memory utilization during predictions to evaluate their resource usage. The models were compiled into individual executables, and then ran on an undisturbed environment, monitoring resource utilization using a resource monitor. CPU usage measures the computational load on the central processing unit during prediction or training.

Table VI

Resource Utilization by Models During Prediction

|  | Shallow20 | Deep20 | Shallow40 | Deep40 | Concat20 | Concat40 | Maximum20 | Maximum40 | Minimum40 |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Prediction Time (S) | 0.65 | 0.82 | 0.47 | 0.54 | 0.70 | 0.86 | 0.70 | 0.73 | 0.70 |
| Max CPU (%) | 0.30 | 0.27 | 0.34 | 0.33 | 0.19 | 0.22 | 0.26 | 0.31 | 0.36 |
| Memory Allocated (GB) | 28.12 | 28.00 | 31.44 | 29.78 | 29.06 | 31.93 | 29.02 | 30.39 | 31.32 |
| Memory Used (MB) | 19.91 | 19.73 | 23.13 | 21.59 | 20.78 | 23.60 | 20.27 | 22.22 | 23.08 |
| Model Size (KB) | 110 | 648 | 190 | 708 | 794 | 794 | 788 | 1086 | 968 |

Resource usage by tested models shows that models with more features tend to have lower prediction times, probably due to the availability of more information.

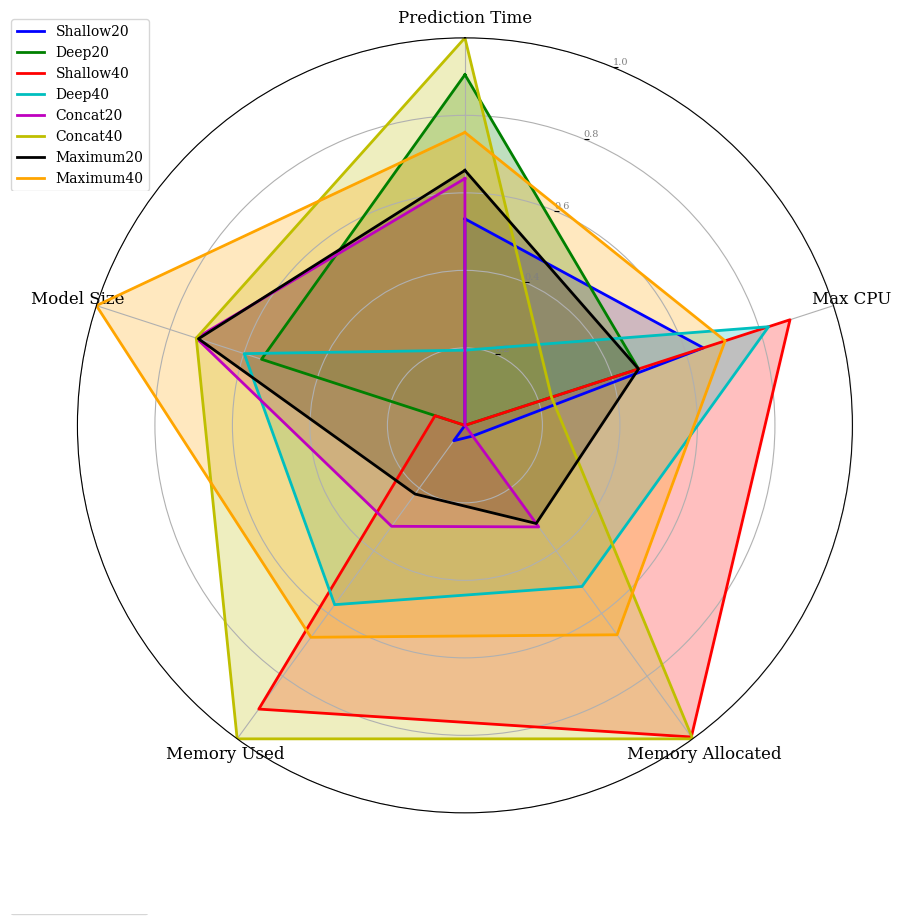


Fig. 20 Resource utilization results of selected models

Based on the radar chart shown in Figure 20, in terms of resource utilization, the Shallow20 individual model uses the fewest resources. Among the 20-feature models, Concat20 performs better, and among the 40-feature models, Maximum40 performs better.

Overall, the individual Shallow20 model has the smallest model size, and lowest inference time, moderate memory and CPU usage, indicating its resource efficiency. Concat20 and Maximum40 demonstrate better resource efficiency although with 794 KB and 788KB model sizes significantly higher than Shallow20 with 110 KB. Interestingly, they are competitive in CPU and memory usage. Among all models, despite the model size, Concat20 offers the lowest memory and CPU usage with 20.78 MB and 0.19% respectively.

In low-resource edge environments, prioritizing smaller model sizes, and lower CPU and memory usage while maintaining acceptable accuracy would be beneficial. Hence Shallow20 model providing a trade-off in acceptable performance with low resource usage and faster inferencing might be appropriate for low-resource edge servers. High-resource environments require prioritizing accuracy, even if it means larger model sizes and resource usage. Hence, Concat20, Maximum40, and Minimum40 are appropriate choices for high-end servers.

Correlation Analysis

Correlation analysis as shown in Figure 21 offers valuable insights about the relationships between various factors and performance metrics. It enables understanding models' behavior, detecting potential overfitting, comparing model variations and relative effectiveness, as well as validating model assumptions. Ultimately, correlation analysis assists in the selection of better models and the optimization of their configurations for better performance.

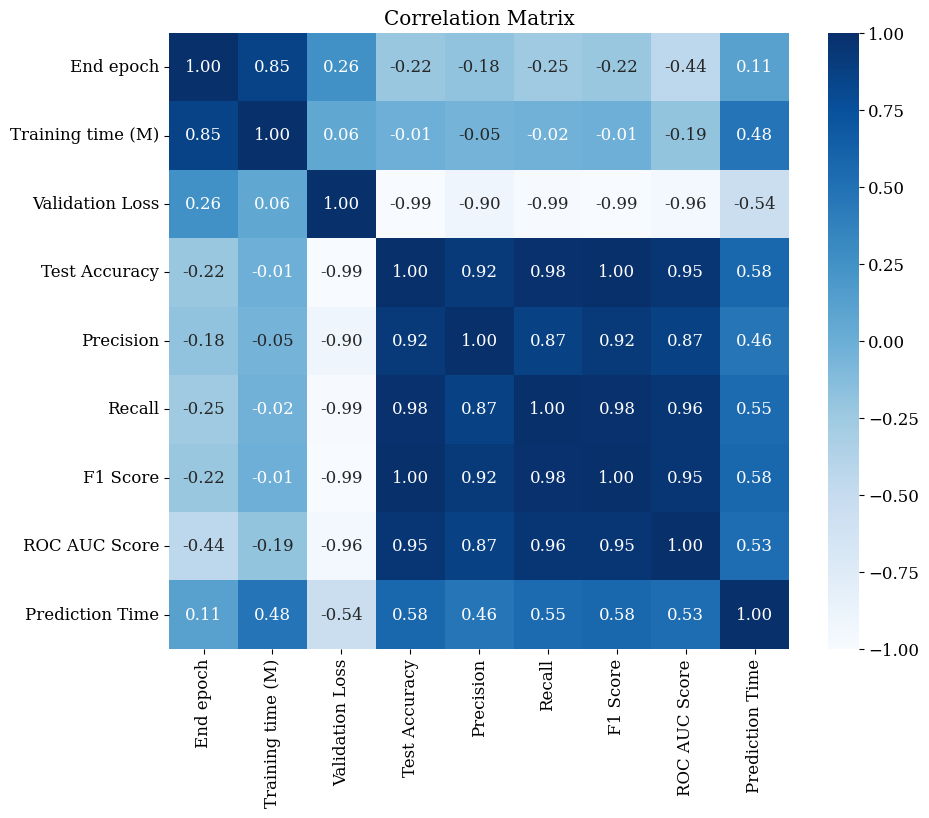


Fig. 21 Correlation analysis of evaluation metrics

Based on correlation analysis, there is a strong positive correlation of 0.85 between the number of epochs for convergence and training time. It indicates that training time increases as the number of epochs increases, as expected. The moderate positive correlation of 0.26 between the number of epochs and the validation loss, suggests that the validation loss tends to increase slightly, along with the number of epochs, indicating possible overfitting with longer training time. Test accuracy, precision, recall, and F1 score show negative correlations ranging from -0.22 to -0.25 with the number of epochs. It implies a tendency for performance metrics to decrease slightly as the number of epochs increases.

The ROC AUC score shows a moderate negative correlation of -0.44 with the number of epochs, suggesting that it tends to decrease as the number of epochs increases. Test accuracy, precision, recall, F1 score, and ROC AUC score show a strong negative correlation ranging from -0.89 to -0.99 with the validation loss. This indicates that as the validation loss decreases, these performance metrics tend to increase significantly. Test accuracy, precision, recall, F1 score, and ROC AUC score are highly positively correlated with each other, ranging from 0.87 to 1.00. This implies that these metrics move together, indicating model performance. There is a moderate positive correlation of 0.48 between the prediction time and the training time. It suggests that models with longer training times may also have longer prediction times. There is a moderate negative correlation of -0.54 between the prediction time and the validation loss, suggesting that models with longer prediction times may have lower validation losses. This analysis suggests that faster converging models tend to offer better performance and lower overfitting. The insights further confirm the suitability of Concat40, Minimum40, and Maximum40 models for higher performance with lower overfitting potential in edge servers network attack detection.

Overall, several observations can be made from the evaluation of the results. All hybrid models generally outperformed the individual Shallow and Deep models in terms of accuracy, precision, recall, F1 score, and ROC AUC score. The results indicate the robust performance of Hybrid models in detecting network attacks on edge servers. While hybrid models offer enhanced detection ability, they also consume longer training times and more epochs to converge, including slightly higher inference times. Among the hybrid models, Concat40, Maximum40, and Minimum40 strike the best balance in evaluated performance metrics. Particularly, the Maximum40 and concat40 hybrid models exhibit strong performance across performance metrics as well as resource usage, further confirmed by correlation analysis. Among those two, the Concat40 model is preferable in terms of computational efficiency, training time, and resource usage for providing optimal performance for federated learning tasks. The Maximum20 and Maximum40 hybrid models are suitable for one-time training tasks. Ultimately, hybrid models hold promise for network attack detection in edge computing environments offering significant improvements over the individual shallow and deep models alone.

Future Direction:

Building on the insights gained from this study on network attacks on edge servers, the investigation seeks to further explore communication security and privacy in AV environments. This expansion will focus on network attack and anomalous behavior detection, addressing adversarial attacks, and preserving privacy in AVs. It will further refine and optimize hybrid models and federated learning approaches for adapting them to the distributed and dynamic nature of vehicular networks to enhance the security and reliability of AVs. The aim of addressing these future directions is to advance the state-of-the-art in autonomous vehicle communications security and privacy, ultimately contributing to the development of safer and more secure autonomous vehicle systems.

1. **Conclusions**

This research demonstrates the efficacy 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 several performance metrics. Among the Hybrid Fusion models, the Maximum20 model, achieving an accuracy of 95.34% and a precision score of 0.98, is suitable for low-resource edge servers. Meanwhile, the Maximum40 model, with the highest accuracy of 98.56% and a precision score of 0.98, is ideal for high-resource edge servers.

The study also highlights the trade-offs between model performance and resource consumption. Hybrid Fusion models using 40 features generally demonstrated higher accuracy and better performance metrics but required more computational resources and longer training times. The Concat40 and Maximum40 models struck a balance between performance and resource efficiency, making them ideal for deployment in high-resource environments or for use in federated learning scenarios where frequent model updates are necessary. Furthermore, the individual Shallow20 model was identified as the most resource-efficient, with the smallest model size, lowest inference time, and low memory and CPU usage, achieving a moderate accuracy of 0.93. This makes it a viable option for ultra-low-resource edge environments demanding low latency over accuracy, where quick deployment and efficient resource utilization are critical.

Overall, the Shallow-Deep Hybrid Fusion models provide a robust solution for network attack detection on edge servers, capable of adapting to varying resource constraints. These findings offer valuable insights for optimizing edge server security and highlight the potential for applying these models beyond cybersecurity to other domains requiring efficient and accurate predictive analytics. Future research aims to explore communication security and privacy in autonomous vehicles, prioritizing low latency, high reliability, and security by optimizing hybrid models and federated learning to ensure the safety of autonomous vehicles.

References

Abreha, H.G., Hayajneh, M. and Serhani, M.A. (2022). Federated Learning in Edge Computing: a Systematic Survey. *Sensors*, [online] 22(2), p.450. doi:<https://doi.org/10.3390/s22020450>.  
  
Aggarwal, C.C. (2018). *Neural Networks and Deep Learning : a Textbook*. Cham: Springer International Publishing.

Ahmed, M. and Dowland, P.H. (2021). *Secure Edge Computing*. CRC Press.

Alwarafy, A., Al-Thelaya, K.A., Abdallah, M., Schneider, J. and Hamdi, M. (2020). A Survey on Security and Privacy Issues in Edge Computing-Assisted Internet of Things. *IEEE Internet of Things Journal*, pp.1–1. doi:<https://doi.org/10.1109/jiot.2020.3015432>.

Caviglione, L., Comito, C., Guarascio, M. and Manco, G. (2023). Emerging Challenges and Perspectives in Deep Learning Model security: a Brief Survey. *Systems and Soft Computing*, p.200050. doi:<https://doi.org/10.1016/j.sasc.2023.200050>.

Choi, D.Y., Kim, D.-H. and Song, B.C. (2020). Multimodal Attention Network for Continuous-Time Emotion Recognition Using Video and EEG Signals. *IEEE Access*, 8, pp.203814–203826. doi:<https://doi.org/10.1109/access.2020.3036877>.

Chollet, F. (2018). *Deep Learning with Python*. Shelter Island (New York, Estados Unidos): Manning, Cop.

Cook, J., Rehman, S.U. and Khan, M.A. (2023). Security and Privacy for Low Power IoT Devices on 5G and beyond Networks: Challenges and Future Directions. *IEEE Access*, 11, pp.39295–39317. doi:<https://doi.org/10.1109/access.2023.3268064>.

David, W. (2019). *UNSW\_NB15 Dataset*. [online] www.kaggle.com. Available at: <https://www.kaggle.com/datasets/mrwellsdavid/unsw-nb15> [Accessed 25 Feb. 2024].

Fazeldehkordi, E. and Grønli, T.-M. (2022). A Survey of Security Architectures for Edge Computing-Based IoT. *IoT*, 3(3), pp.332–365. doi:<https://doi.org/10.3390/iot3030019>.

Ganaie, M.A., Hu, M., Malik, A.K., Tanveer, M. and Suganthan, P.N. (2022). Ensemble Deep learning: a Review. *Engineering Applications of Artificial Intelligence*, 115, p.105151. doi:<https://doi.org/10.1016/j.engappai.2022.105151>.

Gartner (2018). *What Edge Computing Means for Infrastructure and Operations Leaders*. [online] www.gartner.com. Available at: <https://www.gartner.com/smarterwithgartner/what-edge-computing-means-for-infrastructure-and-operations-leaders> [Accessed 14 Feb. 2024].

Goodfellow, I., Bengio, Y. and Courville, A. (2016). *Deep Learning*. MIT Press.

H. Hapke and Nelson, C. (2020). *Building Machine Learning Pipelines : Automating Model Life Cycles with Tensorflow*. Beijing ; Boston ; Farnham ; Sebastopol ; Tokyo O’reilly.

Jasper, S. (2022). *Russian Cyber Operations*. Georgetown University Press.

Javeed, D., Gao, T., Khan, M.T. and Ahmad, I. (2021). A Hybrid Deep Learning-Driven SDN Enabled Mechanism for Secure Communication in Internet of Things (IoT). *Sensors*, 21(14), p.4884. doi:<https://doi.org/10.3390/s21144884>.

Jin, X., Katsis, C., Sang, F., Sun, J., Kundu, A. and Kompella, R.R. (2022). Edge Security: Challenges and Issues. *arXiv (Cornell University)*. doi:<https://doi.org/10.48550/arxiv.2206.07164>.

Jullian, O., Otero, B., Rodriguez, E., Gutierrez, N., Antona, H. and Canal, R. (2023). Deep-Learning Based Detection for Cyber-Attacks in IoT Networks: A Distributed Attack Detection Framework. *Journal of Network and Systems Management*, 31(2). doi:<https://doi.org/10.1007/s10922-023-09722-7>.

Khan, M.A., Khan, N.W., Alshehri, M., Alazeb, A., Ullah, S., Naz, N. and Ahmad, J. (2023). A Hybrid Deep learning-based Intrusion Detection System for IoT Networks. *Mathematical Biosciences and Engineering*, 20(8), pp.13491–13520. doi:<https://doi.org/10.3934/mbe.2023602>.

Kim, H. and Lee, K. (2022). IIoT Malware Detection Using Edge Computing and Deep Learning for Cybersecurity in Smart Factories. *Applied Sciences*, 12(15), p.7679. doi:<https://doi.org/10.3390/app12157679>.

Li, W., Peng, Y., Zhang, M., Ding, L., Hu, H. and Shen, L. (2023). Deep Model Fusion: a Survey. *arXiv (Cornell University)*. doi:<https://doi.org/10.48550/arxiv.2309.15698>.

Liu, Y., Kumar, N., Xiong, Z., Wei Meng Lim, Yu, R. and Dusit Niyato (2020). Communication-Efficient Federated Learning for Anomaly Detection in Industrial Internet of Things. doi:<https://doi.org/10.1109/globecom42002.2020.9348249>.

Meegammana, N.W. and Fernando, H. (2023). Detection of Network Attacks on Application Servers Using Deep Learning in IoT Environments. doi:<https://doi.org/10.1109/icac60630.2023.10417159>.

Meegammana, N.W. and Fernando, H. (2024). Securing IoT Servers: Shallow vs. Deep Neural Network Architectures. *[Un Published]*.

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

Qazi, E.-H., Muhammad Hamza Faheem and Zia, T. (2023). HDLNIDS: Hybrid Deep-Learning-Based Network Intrusion Detection System. 13(8), pp.4921–4921. doi:<https://doi.org/10.3390/app13084921>.

Qiu, T., Chi, J., Zhou, X., Ning, Z., Atiquzzaman, M. and Wu, D.O. (2020). Edge Computing in Industrial Internet of Things: Architecture, Advances and Challenges. *IEEE Communications Surveys & Tutorials*. [online] doi:<https://doi.org/10.1109/COMST.2020.3009103>.

S. Abimannan, El-Alfy, E.-S.M., Chang, Y.-S., Hussain, S., Shukla, S. and D. Satheesh (2023). Ensemble Multifeatured Deep Learning Models and Applications: a Survey. *IEEE Access*, 11, pp.107194–107217. doi:<https://doi.org/10.1109/access.2023.3320042>.

Sagu, A., Gill, N.S. and Gulia, P. (2020). Artificial Neural Network for the Internet of Things Security. *International Journal of Engineering Trends and Technology*, 68(11), pp.129–136. doi:<https://doi.org/10.14445/22315381/ijett-v68i11p218>.

Sagu, A., Gill, N.S. and Gulia, P. (2022). Hybrid Deep Neural Network Model for Detection of Security Attacks in IoT Enabled Environment. *International Journal of Advanced Computer Science and Applications*, 13(1). doi:<https://doi.org/10.14569/ijacsa.2022.0130115>.

Satyanarayanan, M. (2017). Edge Computing. *Computer*, 50(10), pp.36–38. doi:<https://doi.org/10.1109/mc.2017.3641639>.

Sedjelmaci, H., Senouci, S.M., Ansari, N. and Boualouache, A. (2021). A Trusted Hybrid Learning Approach to Secure Edge Computing. *IEEE Consumer Electronics Magazine*, pp.1–1. doi:<https://doi.org/10.1109/mce.2021.3099634>.

Shi, Y., Feng, H., Geng, X., Tang, X. and Wang, Y. (2019). A Survey of Hybrid Deep Learning Methods for Traffic Flow Prediction. doi:<https://doi.org/10.1145/3373419.3373429>.

Sulieman, N.A., Ricciardi Celsi, L., Li, W., Zomaya, A. and Villari, M. (2022). Edge-Oriented Computing: a Survey on Research and Use Cases. *Energies*, 15(2), p.452. doi:<https://doi.org/10.3390/en15020452>.

Sutaria, N. (2022). *Bias and Ethical Concerns in Machine Learning*. [online] ISACA. Available at: <https://www.isaca.org/resources/isaca-journal/issues/2022/volume-4/bias-and-ethical-concerns-in-machine-learning> [Accessed 18 Apr. 2024].

Tariq, U., Ahmed, I., Bashir, A.K. and Shaukat, K. (2023). A Critical Cybersecurity Analysis and Future Research Directions for the Internet of things: A Comprehensive Review. *Sensors*, 23(8). doi:<https://doi.org/10.3390/s23084117>.

Ullah, S., Khan, M.A., Ahmad, J., Jamal, S.S., e Huma, Z., Hassan, M.T., Pitropakis, N. and Buchanan, W.J. (2022). HDL-IDS: a Hybrid Deep Learning Architecture for Intrusion Detection in the Internet of Vehicles. *Sensors*, [online] 22(4), p.1340. doi:<https://doi.org/10.3390/s22041340>.

Wu, Y., Wei, D. and Feng, J. (2020). Network Attacks Detection Methods Based on Deep Learning Techniques: a Survey. *Security & Communication Networks*, [online] pp.1–17. doi:<https://doi.org/10.1155/2020/8872923>.

Xiao, Y., Jia, Y., Liu, C., Cheng, X., Yu, J. and Lv, W. (2019). Edge Computing Security: State of the Art and Challenges. *Proceedings of the IEEE*, 107(8), pp.1608–1631. doi:<https://doi.org/10.1109/jproc.2019.2918437>.

Zeyu, H., Geming, X., Zhaohang, W. and Sen, Y. (2020). *Survey on Edge Computing Security*. [online] IEEE Xplore. doi:<https://doi.org/10.1109/ICBAIE49996.2020.00027>.

Zhang, Y., Liu, Y., Guo, X., Liu, Z., Zhang, X. and Liang, K. (2022). A BiLSTM-Based DDoS Attack Detection Method for Edge Computing. *Energies*, 15(21), p.7882. doi:<https://doi.org/10.3390/en15217882>.