RESEARCH

Unified AI Models for Network Security on Edge Devices

Sengul Bayrak[1] · Alper Karaca[1] · Ferhat Toson[1] · Mehmet Emin Tayfur[1] · Selcuk Yavas[1]

Received: 6 June 2025 / Revised: 15 August 2025 / Accepted: 7 September 2025

Abstract
Rapidly identifying and mitigating security threats to reduce the impact of attacks is one of the most pressing challenges of our time. Digital threats frequently jeopardize users, often manifested as web-based, intranetwork, or spam-related attacks. The literature indicates that most studies examine each type of attack separately. This study introduces six distinct deep learning (DL) model architectures capable of detecting three attacks. The results of this study demonstrate that the Deep Neural Network (DNN) model achieved a test accuracy of 98.78% for the FWAF dataset. The CNN-LSTM model achieved 99.95% accuracy for the KDDCUP-99 dataset and 99.09% accuracy for the SMS Spam dataset. This study accurately identifies anomalous activities and anticipates potential attacks by analyzing large data sets from various data channels. Furthermore, this work improves the efficacy of existing firewalls and holds significant promise in preventing malicious factors from exploiting vulnerabilities in digital infrastructures. This study conducted experiments in a real-time environment using artificial intelligence models deployed on Jetson Nano. For malicious traffic detection, malicious traffic was created on the network with the Nmap tool, and network traffic was monitored using Wireshark, followed by classification using our model. In the Damn Vulnerable Web Application (DVWA) platform, a malicious URL classification model was used to test web attacks. For spam detection, suspicious text was retrieved from the user via a plugin running in the browser and sent to the spam classification model, and the result was returned to the user. In conclusion, DL models have the potential to detect and prevent various security threats in web applications. This study enables the analysis of normal operating behaviors in web applications and the identification of anomalous activities. This capability helps detect attempted intrusions or potential security issues indicating user behavior. Furthermore, the study facilitates the automatic detection of attacks targeting web applications and the implementation of countermeasures to mitigate such threats.

Keywords Network security · Deep learning models · Natural language processing · Smart edge device

1 Introduction

Nowadays, technology is advancing rapidly, and artificial intelligence, an important part of these advances, is effectively used in every aspect of our lives. Traditional security measures are no longer sufficient against complex and dynamic threats. At this point, AI-based security systems provide a significant advantage in detecting and defending against threats. This research aims to counter modern threats from Intrusion Detection System (IDS), Query, and SMS Spam using DL model architectures to improve the security of computer systems. IDS is a security measure used to detect and take measures against potential attacks on a network or system. By monitoring network traffic or system activities, it detects and reports signs of attacks or anomalies. Attacks from the digital world often endanger users in the form of queries, networks, and SMS spam. This study aims to develop a real-time system to

Alper Karaca, Ferhat Toson, Mehmet Emin Tayfur and Selcuk Yavas have contributed equally to this work.

Published online: 20 October 2025

detect these three types of attacks. Studies in the literature have mainly examined each attack separately. Recent studies in the literature on this subject are examined in three separate headings.

Traditional network-based intrusion detection systems are based on signature-based or rule-based methods. These systems identify potential threats by matching network traffic against a database of previously defined attack patterns. While this method is effective for known attacks, it can be inadequate in cases such as zero-day attacks and encrypted traffic. In addition, manually updating and maintaining the rule sets of these systems creates a significant operational burden. False positive rates are also a major problem in high-traffic networks. In contrast, deep learning-based intrusion detection systems offer a more flexible and learnable structure by analyzing features obtained from network traffic. The models used in this study have the ability to learn directly from data without requiring predefined rules, as is necessary in classical systems. This enables the system to detect unknown or variant attacks. Furthermore, when the classification accuracy of these models is increased, false positive rates can also be reduced. While systems such as Snort and Suricata offer powerful solutions in terms of real-time processing capacity, trained deep learning models are attractive because they can be sensitive to a wide variety of attacks with relatively fewer resources and less human intervention. The models deployed on Jetson Nano in this study were able to classify incoming network requests with low latency and produced successful results against different types of attacks. Deep learning-based systems also have some limitations. The representativeness of the data used in the training process directly affects the model's generalization ability. Additionally, the explainability level of modeled systems is lower compared to classical rule-based systems. However, the fact that the datasets used in this study have understandable and modelable characteristics provides an important advantage in terms of explainability and testability.

1.1 Literature Review for IDS

IDS is a security measure that monitors, detects, and reports abnormal activities occurring on a network or system. This system attempts to identify potential cyber-attacks or malicious activities. IDS effectively detects Distributed Network Attacks (DDoS), Man-in-the-Middle (MITM), Remote-to-Local, and U2R attacks. In the literature, studies on DDoS have been conducted in the form of DDoS defense systems, attack capture, attack reaction, and identification of the attack source. Studies on DDoS in the literature have been done in three different ways: attack detection, analysis of an anomaly on the network on incoming packets, and identification of attacks. Resource management uses the attack reaction technique to reduce the impact of attacks. Attack source identification aims to identify the location of the victim site to the attacker, even if the attacker spoofs the Internet Protocol (IP) address. Aktar and Nur [1] studied the Deep Contractive Autoencode (DCAE) model for DDoS attack capture. They achieved an accuracy of 0.934 to 0.975 for the CIC-DDoS2019 dataset for properly reconstructing normal traffic and separating attack data from non-attack data. Denial of service (DOS) attack is one of the main types of attacks that threaten WSNs. Salmi and Oughir [2] developed a DL-based IDS to detect DoS attacks in wireless sensor networks (WSN). Their system includes four types of DoS attacks that affect WSNs, namely Blackhole, Grayhole, Flooding, and Scheduling attacks. They obtained an overall accuracy of 0.969 for these four attacks in model performance with CNN and RNN. Wang et al. [3] proposed a new IDS model and combined Stacked Contractive Autoencoder (SCAE) and Support Vector Machine (SVM) algorithms. This model uses SCAE to extract low-dimensional features from raw inputs automatically. The model's training process consisted of three stages: unsupervised pre-training, unrolling, and supervised fine-tuning. After the training, the SVM classifier detected anomalous examples from the extracted features. To evaluate the intrusion detection performance of the model, the researchers conducted some experiments on two datasets, NSL-KDD and KDD99. The approach achieved 0.887 accuracy on binary classification tasks on the NSL-KDD dataset. Kim et al. [4] developed a DL-based intrusion detection model for DoS attacks using KDDCUP-99 and CSE-CIC-IDS2018 datasets. They created two types of attack images in RGB and grayscale. They designed a Convolutional Neural Network (CNN) model by considering parameters such as the number of layers and kernel size used. While the CNN model provided over 0.990 accuracy in binary and multiclass classifications in the KDD dataset, the RNN model showed 0.990 accuracy

in binary classification and 0.930 in multiclass classification. In the CSE-CIC-IDS 2018 dataset, the CNN model achieved an average accuracy of 0.915, while the RNN model achieved an average accuracy of 0.65. Ghanem and Erbay [5] proposed to build a new model based on deep contextualized word representation. As a result, in this work, they modeled a new DL architecture, CBLSTM (Contextualized Bidirectional Long Short-Term Memory neural network), based on a bidirectional long short-term neural network with embedding from language models to address the spam text problem in social networks. They obtained an accuracy of 97.80 with Twitter data, 0.952 with YouTube data, and 0.993 with SMS data. Altuncu (2021) [6] modeled the NSL-KDD dataset with multiple classification models and classified each type of attack with 0.853 accuracy. Demir and Aslan (2023) [7] modeled the Fırat University Firewall Device with 0.984 accuracy with the K-nearest neighbor method. Özekes and Karakoc (2019) [8] worked with NSL-KDD and KDD-Cup datasets and developed models with decision trees and random forest methods. They obtained an accuracy rate of 0.99 as a result of both modeling. Demir (2021) [9] worked with ISCX-2012 dataset in his study. In this study, 100% accuracy with the decision tree was. Sahingoz et al. (2019) [10] developed modeling with 0.951 accuracies with k-nearest neighbor modeling, 0.915 for decision trees, and 0.962 accuracy with the ADA method using the NSL-KDD dataset. Seyyar et al. (2022) [11] modeled a web intrusion detection system that addresses the security threats that arise with the increasing use of web applications in all areas and the increase in attacks against web applications. The web intrusion detection system is a model that can distinguish between normal and abnormal URL. They used the BERT model in the URL analysis phase. They used the CNN model in the classification phase. CSIC 2010, FWAF, and HTTP Params data for training and testing achieved an accuracy of over 0.960. Ilhan (2019) [12], using Yahoo Webscope S5 and DARPA 1998 datasets, it was observed that the C-LSTM model and NSA-assisted method achieved an average of 0.943 in correctly finding anomalies in web traffic data and an average of 0.977 in the overall classification rate. Gul (2018) [13], using the UNB ISCX IDS 2012 dataset, using SDG, DT, KNN, and Artificial Neural Network (ANN) models for the detection of insider attacks with the network behavior model, accuracy rates of 0.950, 0.9601, 0.965, 0.984, respectively, were determined. Ahmetoglu et al. (2021) [14] propose an approach based on feature selection and machine learning for web application attack detection using CIC-IDS2017 and CSE-CICIDS2018. Using Random Forest (RF), SVM, Naive Bayes (NB), KNN, and DL algorithms, kappa scores of 0.970, 0.870, 0.300, 0.950, and 0.960 were obtained respectively. Yu et al. [15] obtained a recall value of 0.9830 by modeling CICIDS and CSECICIDS datasets with an autocoder using the Packet Bytes-based Convolutional Neural Network (PBCNN) method.

1.2 Literature Review for Query Attack

Web attacks are malicious attempts on websites or web applications. Stealing information and creating service interruption are the main objectives of these attacks. SQL Injection, XSS, DDoS, Brute Force Attacks, and Phishing are some of these attacks. Methods such as Machine Learning and Artificial Intelligence, AI Supported Bot Blocking, and Wireless Application Firewall (WAF) are used to prevent these attacks. The main function of WAF is to control and filter requests to web applications and block malicious traffic. In this way, it helps to limit potential attacks, malware, data leaks, and other web application vulnerabilities. A WAF inspects and filters HTTP requests according to specified rules, blocks malicious traffic, monitors users' authentication processes, and implements authorization controls. Brown et al. [16] presented a comprehensive analysis and insights on using Automated Machine Learning (AutoML for static, dynamic, and online malware detection. For static analysis, they used SOREL-20 M to demonstrate effectiveness on large datasets and the smaller EMBER-2018 dataset, which was specially prepared to challenge the performance of machine learning models. They used CNN for cloud IaaS in online malware detection scenarios with AutoML. They achieved 0.998 with SOREL-20 M dataset and 0.984 with EMBER-2018 dataset. In the online malware capture, they achieved 0.989 accuracy with the most accurate model Darts AutoML in Baseline. In Application, they achieved 0.986 accuracy with 7-layer Darts AutoML. Web applications have become the center of the digital environment, providing users with instant access to information and allowing businesses to expand their reach. Injection attacks such as SQL injection (SQLi)

are prominent attacks against web applications, given that most applications integrate a database system. Paul et al. [17] formulate the SQLi attack as a multi-class, multi-attack vector, prioritization and prevention problem. They collected 457.233 benign and malignant network traffic samples and 70.023 samples with SQLi and benign payloads for attack detection and classification. After evaluating several machine learning-based algorithms, the hybrid CNN-LSTM models achieved an average F1 score of 0.97 on the web and network. Dawadi et al. [18] conducted a comparative analysis between normal HTTP traffic and attack traffic to protect against threats such as DDoS, SQL injection and XSS. They examined various features from ISCX, CISC, and CICDDoS standard datasets and identified the differences between normal and attack traffic. In the study, a layer architecture model was developed to detect DDoS, XSS and SQL injection attacks. The first layer of this LSTM-based architecture, created with the data collected from the simulation environment, was designed as a DDoS detection model with an accuracy rate of 0.976. The second layer is designed for XSS and SQL injection detection with an accuracy rate 0.893. High HTTP traffic was first inspected, filtered and then forwarded to the second layer. WAF has contributed to the literature by adding an additional layer of security to web applications by filtering at the application level, which traditional network firewalls cannot provide. Koksal et al. [19] used ensemble learning methods and RF algorithm to classify benign and malignant URLs using the URL database of the Canadian Institute for Cyber Security (ISCX-URL-2016). The results show binary classification success with an average accuracy of 0.9942 and multiclass classification success with an average accuracy of 0.9568. Ucar et al. [20] developed a DL model for detecting malicious URLs on the ISCX-URL-2016 dataset. They used LSTM and CNN's DL algorithms to detect benign and malicious URLs. According to the experimental results, LSTM, with an accuracy rate of 0.91, and CNN, with an accuracy rate of 0.95, achieved significant success detecting malicious web pages. Abdi and Wenjuan [21] conducted experiments on 344.821 benign URLs and 75.643 malicious URLs, using a simple algorithm to detect malicious URLs with an accuracy of more than 0.96. In this algorithm, machine learning techniques were used instead of traditional blacklisting. Tiryaki et al. (2023) [22] created a big dataset by combining two similar national and international datasets using a 7-layer RNN model and achieved an accuracy of over 0.91 in detecting malicious URL addresses. Moghimi and Varjani (2016) [23] used an Artificial Neural Network (ANN) model, specifically the Multilayer Perceptron (MLP) type, to classify websites with phishing characteristics. The results show that this model effectively classifies such websites with a accuracy rate of 0.9914. Ismail et al. [24] present an in-depth evaluation of 19 traditional machine-learning techniques for detecting web application attacks. They developed a model with Neural Networks on refined datasets derived from the HTTPCSIC 2010 dataset with a accuracy rate of 0.90.

1.3 Literature Review for Spam Attack

Spam is unnecessary and inappropriate messages sent electronically to many recipients. It can include advertisements, bogus offers, and malicious software, and it can appear on platforms such as email and phone. Currently, sophisticated algorithms and machine learning techniques are used to detect spam. For example, firewalls and filters block spam content using specific criteria on email servers or client-side software. These criteria can include IP addresses, text content, sender identity, etc. In addition, advanced text analysis and natural language processing techniques are used to deeply analyze text content. Features such as word structure, sentence structure, and language usage are evaluated to detect spam content. It is very important to block deceptive news as spam through tools such as social media, e-mail, and sms. However, extracting relevant features from social networks is difficult due to privacy and time constraints. Contributing to the literature on this topic, Rao et al. [25] presented a hybrid approach to detect social spam using imbalanced social network data to detect SM spam, Ling spam, and Twitter fake news. They used NearMiss and SmoteTomek sampling for dataset balancing. For two-word representation techniques, Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF), they fed Global Vectors for Word Representation (GloVe) pre-trained word embeddings and the second FastText pre-trained word embedding into the same hybrid sequential model. They obtained an accuracy of 0.961 with Glove and 0.973 with FastText. Lai et al. [26] applied TF-IDF and Word2Vec methods to the da aset obtained from fake news on Kag-

gle. They obtained an accuracy of 0.850 with neural networks. Zavrak and Yilmaz [27] modeled detecting spam e-mails using TREC Public Spam, GenSpam, SA, Enron-Spam, and LS datasets with Fast Text and Hierarchical Attentional Hybrid Neural Networks. Five different data sets achieved an average accuracy of 0.993. Ramya and Eswari [28] proposed the Simple Shallow Perceptron Network (SSPN) model to reduce the complications of Deep Neural Networks for high dimensional data, such as high training time, fatigue, and overthinking during network training. The effectiveness of the proposed SSPN model for false news is investigated and identified using the LIAR dataset and the more recent FakeNewsNet dataset. The proposed model outperformed all existing models, with an F1 score of 0.730 for the LIAR dataset and 0.90 for the FakeNewsNet dataset. Karamollaoglu (2019) [29] used the Vector Space Model for spam detection on Turkish and English tweets and obtained a accuracy rate of 0.92 for English and 0.97 for Turkish tweets. Furthermore, Naive Bayes and Random Forest methods implemented using WEKA were observed to be the most successful, with accuracy rates of 0.93 and 0.94, respectively. Tekerek (2019) [30] proposed a technique for detecting spam SMS using SVM methods using the SMS Spam Collection dataset. The results show that the SVM algorithm is the most effective method, with a accuracy rate of 0.983 and a false positive rate of 0.087. The aim of this study is to develop a firewall application using DL models to improve security against web applications, SMS Spam, or IDS-based attacks and to provide more effective protection against cyber attacks. In this direction, an intrusion detection system that can detect attacks on web applications using DL models has been developed. The accuracy and performance of the application will be optimized, tested in real web applications, updated against current attack techniques, and presented to the users with a user-friendly interface. The research results will be shared with the academic and industrial communities to highlight the impact of DL-based firewalls on the security of web applications. The rest of this paper is organized as follows. Section 2 details the DL models used in this related work and details the performance metrics used, as well as the deployment of the developed DL model in the real-time environment. Section 3 presents the experimental results obtained with the DL-based models. Section 4 concludes the future goals of the study and limitations.

2 Materials and Methods

It is based on the binary classification of 3 different data types: malicious Query, IDS, and SMS Spam. In this study, the KDDCUP-99 dataset [32] for IDS, the FWAF dataset [31] for Query, and SMS Spam dataset [33] were used. The first step was data preprocessing and processes such as data normalization, data coding, and feature selection. A deep neural model was applied in the second step, and the preprocessed data was classified. In the third step, the trained models were implemented as a system for testing in real time. The block diagram of the model proposed in this study can be seen in Fig. 1.

2.1 Feature Extraction

It is a process used to extract important information from data sets. It transforms complex data in the data set into simple and meaningful features that can be analyzed.

Data Preprocessing is the step where the data set is cleaned, transformed, and prepared before it is given as input to the model. This stage processes missing or noisy data, standardizes the features, and puts the data set in a suitable format for the model. In this study, since the data came from three different sources, a preprocessing method was applied appropriate to the nature of each data. A data scaling method was applied to bring the numerical values in different ranges in the KDDCUP-99 data set to the same ranges, as shown in Fig. 2. After extracting features from the FWAF dataset, data scaling was applied to bring these numerical values into the ranges we applied to the KDDCUP-99 dataset [32]. The applied normalization process and its impact on the data distribution are illustrated in Fig. 2.

This data was converted into numeric vectors to train DL architectures using text data for SMS Spam detection. The word embedding approach was used for the SMS spam detection.

 https://doi.org/10.1007/s44196-025-00990-6

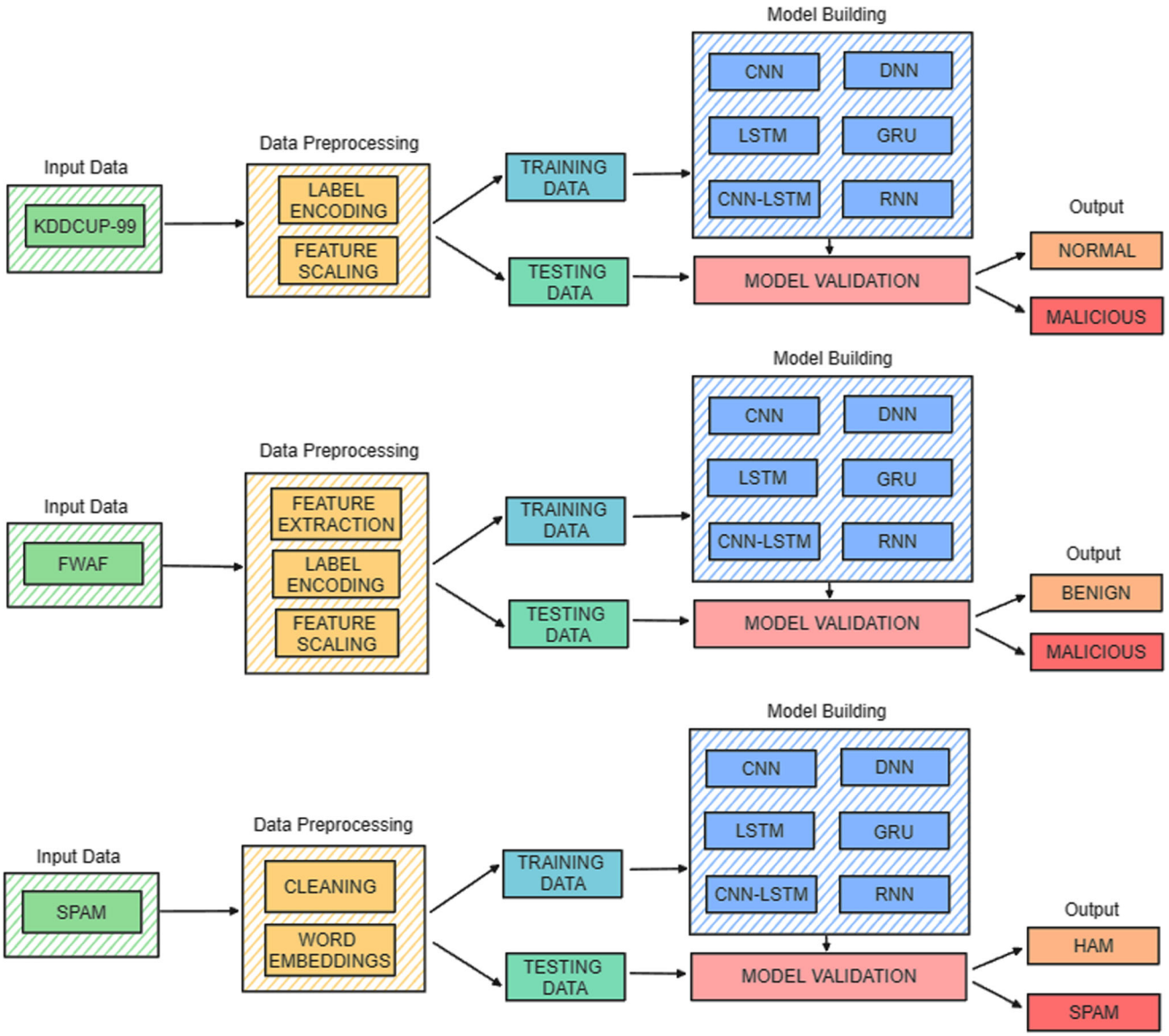

Fig. 1 End-to-end workflow for intrusion detection, malicious url detection and spam detection; showing the sequential stages of data processing, model development, validation, and final prediction output

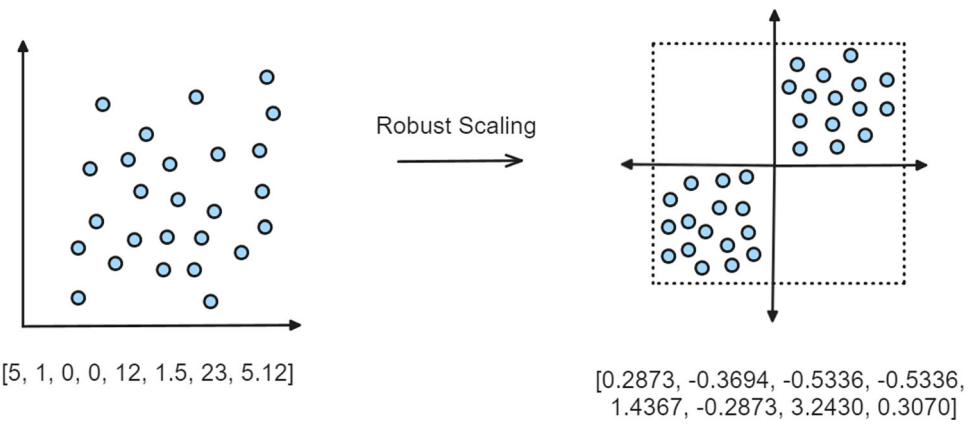

Fig. 2 Illustration of data normalization applied to the KDDCUP-99 dataset, showing the feature values before and after scaling to a common range

2.1.1 Word Embeddings

It is embedded in a way that reflects the semantic similarity of the words. In this way, words are converted into numerical data that the model can process while preserving their semantic similarity. There are two types: Context-unaware and Context-aware. Context-unaware Word Embeddings assign a vector representation of each word fixedly. The representation of each word does not consider any information about its usage in the text, it does not consider the word's environmental context. Context-aware word embeddings consider the environmental context for each word's representation. The representation of each word is linked to the usage of that word in the text and dynamically changes, taking this usage into account. In particular, each word is assigned a representation that considers environmental information, such as its position in the text and the other words with which it interacts. Attention mechanisms are often used for this process [34, 35]. It is one of the word embedding methods. GloVe represents words as vectors, which allows us to better calculate the relationships between words and their meanings. Since GloVe focuses on the context of words, it uses the statistical properties of words to represent them in a high-dimensional space. GloVe calculates the relationships between words based on the statistical information of the word pairs in which the words co-occur. This involves creating a matrix based on word frequencies. This matrix shows the frequency with which a word co-occurs with other words. GloVe evaluates the frequency of co-occurrence of words globally. This helps to understand how often a particular word is found with other words and the meaning of these relationships. GloVe defines a loss function based on the occurrence frequency of word pairs. This function is optimized in the computation of word vectors [36].

2.2 DL Model Architectures

In this study, six different DL architectures were used separately for data from three different data sources.

2.2.1 CNN

It uses convolution and pooling layers to determine the features of the data. In this way, it automatically learns hierarchical representations of features from the data. The convolutional layer is used for feature extraction. Filters are included in this layer. The values in the kernel are shifted over the data, and the values are multiplied by the values of the data. The resulting values are added. This process is applied to the whole data as shown in Fig. 3a.

Activation Function, activation is applied after each convolution layer. This helps the network to learn more complex and general features. The Pooling Layer in Fig. 3b is used to reduce the size of the feature maps and further speed up the network.

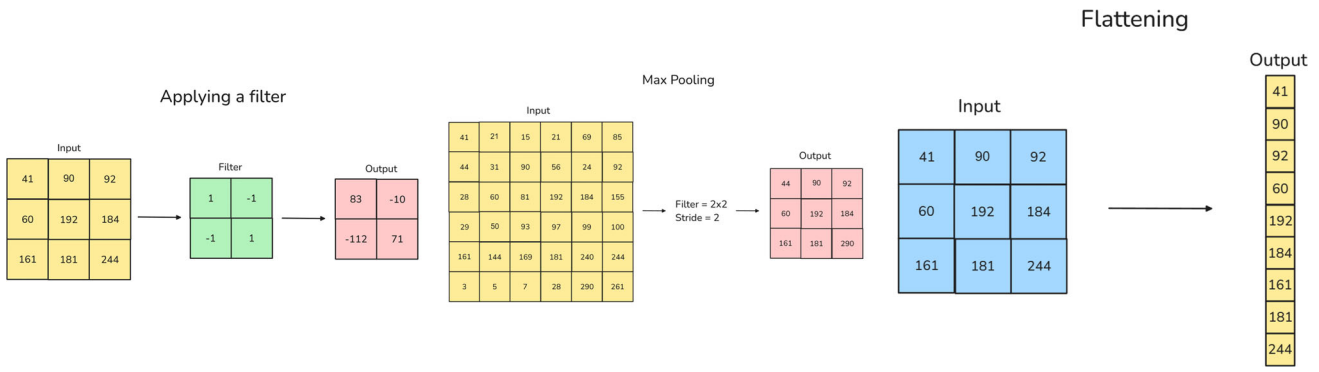

Fig. 3 Key processing steps in the proposed CNN architecture: **a** Feature map generation through learned filters, **b** Dimensionality reduction via max-pooling, and **c** Flattening operation for transition to fully-connected layers. The sequence demonstrates how raw input data is progressively transformed into higher-level representations for classification

The Flattening Layer converts the data into 1×1 as shown in Fig. 3c. Thus, it is prepared for the input of the fully connected data layer. In the fully connected layer, the outputs are passed through this layer to be classified. These layers flatten the feature maps and pass them through an activation function to produce the classification results. The most important feature of CNN is that they can automatically extract input features. However, it is computationally intensive and requires hyperparameter tuning [37].

2.2.2 LSTM

In artificial neural networks (ANN), weight updates are performed by taking the gradients of the error value calculated by the network. However, in deep neural networks, 'vanishing gradient' or 'exploding gradient' problems may arise during backpropagation [38]. These problems arise as a result of gradient values decreasing or increasing exponentially as they propagate between layers. One of the proposed solutions to such problems is the use of smoothed linear units as the activation function.

Long Short-Term Memory (LSTM) networks are a special type of recurrent neural network (RNN) designed to solve this problem. The key difference in LSTMs is that they include a 'memory' mechanism that can carry information between time steps. Each LSTM cell consists of the following components:

1. **Gates (with Sigmoid Activation)**
   - Forget Gate (f_t)
   - Input Gate (i_t)
   - Output Gate (o_t)
2. **Memory States (with Tanh Activation)**
   - Candidate State ($\hat{C}_t$)
   - Memory State (C_t)
   - Hidden State (h_t)

The LSTM network shown in Fig. 4 represents the processing flow at time step t. Mathematically, LSTM operations are expressed by Eqs. 1–9:

- **Forget Gate** (Eq. 1): Determines how much information from the previous memory state will be retained (0 = completely forget, 1 = completely retain).
- **Candidate Memory State** (Eq. 2): Generates a new candidate content $\bar{C}_t$ to be potentially added to the cell state.
- **Input Gate** (Eq. 3): Controls how much of the candidate memory state will be added to the cell state.
- **Output Gate** (Eq. 4): Determines how much information from the memory state is transmitted as output.
- **Memory Update** (Eqs. 5–8): The previous memory state (C_{t-1}) is first multiplied by the forget gate to discard irrelevant information (7), then updated by adding the input gate output and the candidate memory state (5, 8).
- **Hidden State Calculation** (Eqs. 6 and 9): The output gate and the memory state filtered by tanh form the hidden state (H_t) to be transferred to the next time step.

$$f_t = \sigma(X_t * U_f + H_{t-1} * W_f) \tag{1}$$
$$\bar{C}_t = tanh(X_t * U_c + H_{t-1} * W_c) \tag{2}$$
$$I_t = \sigma(X_t * U_i + H_{t-1} * W_i) \tag{3}$$
$$O_t = \sigma(X_t * U_o + H_{t-1} * W_o) \tag{4}$$
$$C_t = f_t * C_{t-1} + I_t * (\bar{C}_t) \tag{5}$$

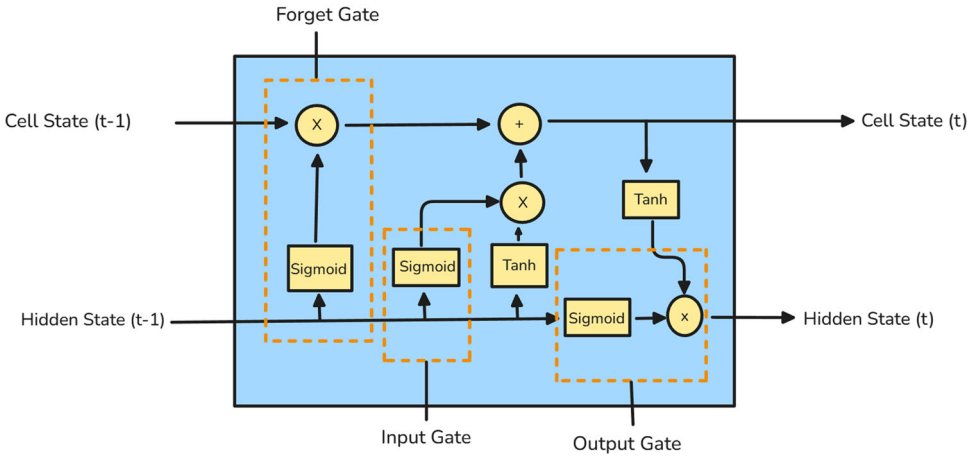

Fig. 4 LSTM cell architecture at time step t illustrating the gate mechanisms (input, forget, output) and cell state operations that enable controlled information flow and long-short term memory

$$H_t = O_t * tanh(C_t) \tag{6}$$
$$C_t = C_{t-1} * f_t \tag{7}$$
$$C_t = C_t + (I_t * \bar{C}_t) \tag{8}$$
$$H_t = tanh(C_t) \tag{9}$$

2.2.3 CNN-LSTM

Combining the algorithmic power of CNN and LSTM architectures provides a powerful method for learning both semantic features and sequential relations of text data to obtain more powerful results. Text data is converted into numerical vectors using word embedding methods such as Word2Vec, GloVe. It applies convolution layers on the text data with CNN architecture. These layers use filters to capture local relationships between words. The convolution layers are followed by pooling layers that help summarize important features and reduce the computational burden of the model. The LSTM component sends features extracted from the CNN with temporarily links on the LSTM layer as sequential data. LSTM enables the model to understand the context by learning the sequential relationships of words and phrases in the text. LSTM preserves the overall meaning and context of the text by retaining information from previous words. Ultimately, the data processed by the CNN and LSTM layers is used for classification through the fully connected layer [38, 39].

2.2.4 GRU

The Gated Recurrent Unit (GRU) is an effective architecture for recurrent neural networks (RNNs) designed as a simplified variant of LSTM. GRUs are computationally more efficient than LSTMs because they require fewer tensor operations and generally offer faster training times [40]. However, despite this simplification, they are quite successful at learning long-term dependencies.

The basic components of GRU consist of two main gates:

1. **Reset Gate** (r_t):
   - Controls how much of the previous hidden state (h_{t-1}) will be forgotten.
   - Regulates the amount of information to be combined with the new input.
   - Uses a sigmoid activation function (values between 0 and 1).
2. **Update Gate** (z_t):

 https://doi.org/10.1007/s44196-025-00990-6

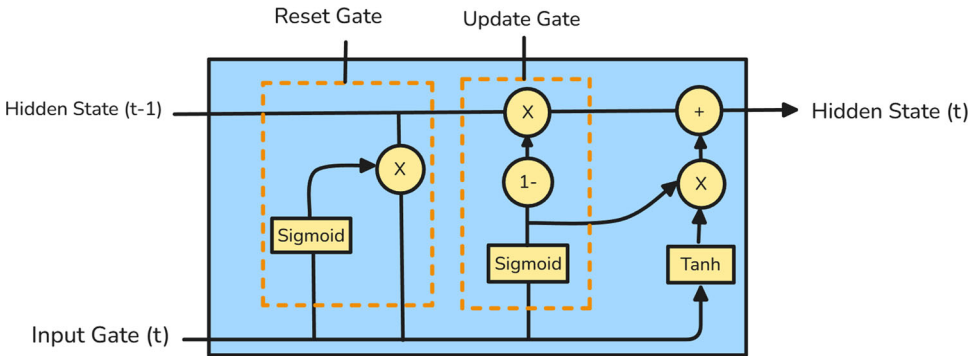

Fig. 5 GRU neural network unit structure illustrating the key components: update gate, reset gate, and hidden state transformation through gating mechanisms

- Determines how much of the previous hidden state (h_{t-1}) will be retained.
- Controls how much of the newly calculated state will be used.
- Again, the sigmoid activation function is applied.

Key differences between GRU and LSTM:

- Uses a single hidden state vector (h_t).
- No separate memory state (cell state).
- Contains fewer parameters (3 weight matrices).
- Uses a reset gate in candidate state calculation.

As shown in Fig. 5, the simpler structure of GRU makes it particularly advantageous for shorter sequences or in situations where computational resources are limited. However, when it comes to very long-term dependencies, LSTMs generally perform better.

2.2.5 RNN

It is a type of artificial neural network specifically designed to process time-dependent or sequential data sequences. These networks have a memory, forming loops that recall information from previous inputs. This property makes them ideal for many tasks, such as language models, time series prediction, and text generation. However, RNNs are known to struggle to handle long-term dependencies. These problems are often called "vanishing gradients" and "exploding gradients" and can be encountered during training. The vanishing gradients problem is when gradients become smaller or disappear during the training of neural networks. As the backward gradients are updated, they become very small as they move towards the lower layers of the model. Therefore, the lower layers cannot be updated sufficiently, negatively affecting the training process. Some activation functions, such as sigmoid and $tanh$, shrink the gradients at certain intervals. Such activation functions cause the problem of vanishing gradients. To overcome this problem, the ReLU activation function is used. ReLU converts gradients to zero for negative inputs. This prevents negatives from taking very small values [41].

2.2.6 DNN

DNN consists of multiple layers and utilizes depth to learn complex data. Text data is converted into numerical vectors using word embedding techniques such as Word2Vec, GloVe, and FastText. This better represents the meaning of words and their relationships. The DNN consists of several hidden layers. These layers are used to extract more complex features from the input data. In each layer, weights and connections between neurons are calculated. The activation of neurons is determined using activation functions. The first layers learn the basic features of words, while deeper layers learn more abstract and complex features. The output layer is used for

classification. The output layer and the softmax activation function are used to calculate the probability distribution between classes. DNNs are trained with a back-propagation algorithm. In this process, the error between the model's predictions and the actual labels is calculated, and this error is used to update the weights. In this way, the model learns the patterns in the data [42, 43].

2.3 Evaluation Metrics

The accuracy and reliability of each DL algorithm developed will be calculated to select the most successful DL architecture by utilizing six different DL modeling methods. Within the scope of this project, precision, recall, F-measure, and accuracy metrics will be calculated for the performance evaluation of the classification model. The values of these metrics will be calculated according to Eqs. 10–13 using true positive (TP), false positive (FP), false negative (FN) and true negative (TN) values in the confusion matrix [44].

$$\text{Accuracy} = \frac{\text{TN} + \text{TP}}{\text{TP} + \text{FP} + \text{TN} + \text{FN}} \tag{10}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{11}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{12}$$

$$\text{F1} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{13}$$

2.4 Wireshark

It is free and open-source software for network analysis, diagnostics, and packet capture. Wireshark is used to monitor and analyze data traffic transmitted on the network. This software is also used in this study as it has powerful tools to visualize the communications taking place in the network, examine packets, and diagnose any problems in the network [45].

2.5 System Implementation

This study used Python language and sci-kit-learn, TensorFlow, and Keras libraries to develop the model. Matplotlib, seaborn, and plotly libraries were used to generate the graphs. These models will be run on NVIDIA Jetson Nano. Jetson Nano is a powerful device designed to support DL and AI applications. Jetson Nano's small size and low power consumption make it easy to use on embedded systems and portable devices. An API was developed with Flask, and the prediction results were displayed in a web interface built with Vue.js, HTML, CSS, and Bootstrap. Packet information was captured with the 'shark' tool on the devices; the addresses of the websites that the user entered and spam messages were captured with the browser plugin to be developed. These captured devices were sent to the Flask API on the Jetson Nano in the network. These prediction results were also displayed on the web interface. Each incoming request is optionally stored on a cloud or Jetson Nano [46].

3 Experimental Analysis

In this study, the KDDCUP-99 dataset [32] for IDS, the FWAF dataset [31] for malicious URLs, and the SMS Spam dataset [33] were used.

 https://doi.org/10.1007/s44196-025-00990-6

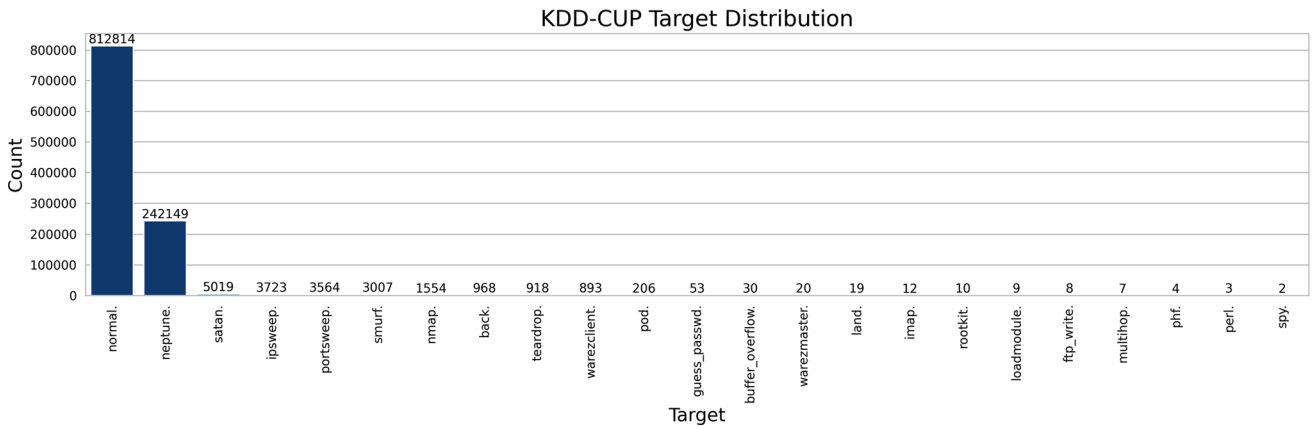

Fig. 6 Label distribution across 23 classes in the KDDCUP-99 intrusion detection dataset, illustrating the significant imbalance between normal connections (majority class) and various attack types

3.1 Datasets

3.1.1 KDDCUP Dataset

KDD-99 is the dataset used for the Third International Competition for Knowledge Discovery and Data Mining Tools, organized with the Fifth International Conference on Knowledge Discovery and Data Mining. The competition task is to build a network intrusion detector, a predictive model that can distinguish between "bad" and "good" normal connections, which are so-called intrusions or attacks. This database contains a standardized dataset to be audited, containing a wide range of intrusions simulated in a military network environment. The label distribution of the KDDCUP-99 dataset is as follows. There are 23 different labels, as shown in Fig. 6. 75.61% of the dataset is Benign, and 24.39% is Malicious.

3.1.2 Query Dataset

The FWAF dataset for malicious queries consists of two files, "goodqueries.txt" and "badqueries.txt". These two files contain 1.310.651 queries. Of these, 44,662 are malicious, and the remaining 1.265,989 are benign. 96.59% of the dataset is Benign, and 3.41% is Malicious. By combining the datasets, a column called "label" was added. The query is labeled with the value "1" if it is malicious and "0" if it is benign.

3.1.3 SMS Spam Dataset

This study was focused on messages received on cell phones. As shown in Fig. 13, the SMS spam dataset consists of 4,825 legitimate SMS messages (86.59%) ("Raw") and a total of 747 (13.41%) spam messages.

3.1.4 Handling Imbalance Datasets

In this study, a systematic hyperparameter tuning process was carried out using the 'keras-tuner' library to configure deep learning models to deliver the best performance. Four key parameters were considered in the optimisation process: optimisation algorithm, learning rate, number of epochs, and batch size. Among the performance metrics, the F1 score was specifically selected because this metric balances precision and recall in imbalanced datasets, thereby more reliably reflecting the model's real-world performance. Hyperparameter combinations were tested to maximize the F1 score during training.

Table 1 Class weight distribution across datasets calculated using Eq. 14

Dataset name	Minority class	Majority class	Weight (minority)	Weight (majority)
KDDCUP-99	Attack	Normal	1.43	0.81
FWAF	Malicious	Benign	3.22	0.77
SMSSpamCollection	Spam	Benign	1.93	0.75

Table 2 Dataset specifications used in experiments, including class distributions and train-test partitions for the three security domains: network intrusion detection (KDDCUP-99), malicious URL detection (FWAF), and spam detection (SMSSPamCollection)

Dataset	KDDCUP-99		FWAF		SMS Spam	
	Benign	Mal.	Benign	Mal.	Ham	Spam
Training Dataset	731,532	235,960	1,139,389	40,196	4,342	672
Test Dataset	81,282	26,218	126,600	4,466	483	75

The datasets used in the three problems addressed in the study (Network Intrusion Detection, Malicious URL Detection, Spam Detection) contain significant uncertainties between classes:

- In the **KDDCUP-99** dataset, the attack/normal traffic ratio is 1:3.10.
- In the **FWAF** dataset, the malicious/benign URL ratio is 1:20.
- In the **SMSSpamCollection** dataset, the spam/ham text ratio is 1:6.46.

To eliminate these imbalances, square root-based inverse frequency weighting was applied to all problems. In order to increase the effect of minority classes on the loss function, the weight of each class was calculated using the formula in Eq. 14:

$$\text{Weight}_{class}(i) = \sqrt{\frac{T}{2T_i}} \tag{14}$$

T represents the total number of samples, and $T_i i$ represents the number of samples in class i. The use of square roots has softened the excessive punishment of the majority class compared to the standard inverse frequency method, while ensuring that minority classes are learned in a balanced manner. Table 1 summarizes the weight values calculated for each data set.

Thus, by increasing the impact of classes with less data, models are able to learn these classes better.

3.2 Feature Extraction

The dataset used in this study is given in Table 2.

The data given in Table 2 is divided into two groups training and test sets; the training set represents 90% of the data set, and the test set represents the remaining 10% of the training set.

3.2.1 Feature Extraction from IDS Data Set

Table 3 shows sample columns and data types in the KDDCUP-99 dataset.

3.2.2 Feature Extraction from Query Data Set

In Figs. 7 and 8, various features are extracted from URL addresses in the FWAF dataset to classify them as benign and malicious.

 https://doi.org/10.1007/s44196-025-00990-6

Table 3 Feature description of the KDDCUP-99 dataset: Column names, their semantic meanings in network traffic analysis, and corresponding data types (continuous, discrete)

The name of feature	Description	Data type
Duration	Length (number of seconds) of the connection	Continuous
protocol_type	Type of the protocol, e.g. TCP, UDP, etc.	Discrete
Service	Network service on the destination, e.g., HTTP, telnet, etc.	Discrete
src_bytes	Number of data bytes from source to destination.	Continuous
dst_bytes	Number of data bytes from destination to source.	Continuous
Flag	Normal or error status of the connection	discrete
Land	1 if the connection is from/to the same host/port; 0 otherwise	Discrete
wrong_fragment	Number of 'wrong' fragments	Continuous
Urgent	Number of urgent packets	Continuous
Hot	Number of 'hot' indicators.	Continuous
num_failed_logins	Number of failed login attempts	Continuous
logged_in	1 if successfully logged in; 0 otherwise	Discrete
num_compromised	Number of 'compromised' conditions	Continuous
root_shell	1 if root shell is obtained; 0 otherwise	Discrete
su_attempted	1 if 'su root' command attempted; 0 otherwise	Discrete
num_root	Number of 'root' accesses	Continuous
num_file_creations	Number of file creation operations	Continuous
num_shells	Number of shell prompts	Continuous
num_access_files	Number of operations on access control files	Continuous
num_outbound_cmds	Number of outbound commands in an ftp session	Continuous
is_hot_login	1 if the login belongs to the 'hot' list; 0 otherwise	Discrete
is_guest_login	1 if the login is a 'guest' login; 0 otherwise	Discrete

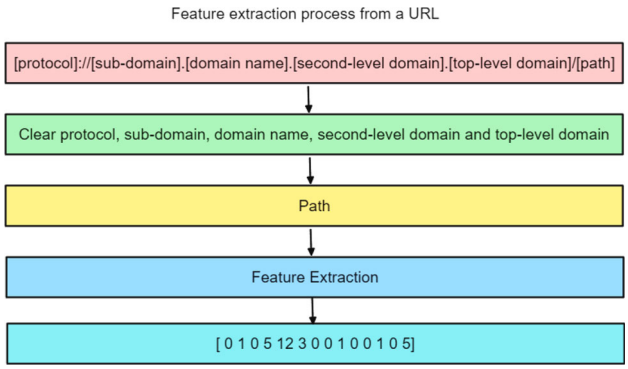

Fig. 7 Feature extraction process from URLs, illustrating the step-by-step transformation from raw URL strings to machine-learning ready features

3.2.3 Feature Extraction from SMS Spam Data Set

GloVe is an unsupervised learning algorithm for obtaining vector representations for words, where training is performed on word-word occurrence statistics from a corpus, and the resulting representations represent a linear subspace of the word vector space. This study uses GloVe 6b 300-dimensional version [37] for embedding. Figure 9 shows the features obtained from the SMS SPAM dataset.

The DL models were applied separately to the datasets obtained after feature extraction.

3.3 Set the Fine Tuning Parameters

The IDS, Query, and SMS-SPAM feature sets were applied to the DL models separately. The optimum parameter values obtained according to the experimental studies are given in Table 2 (Table 4).

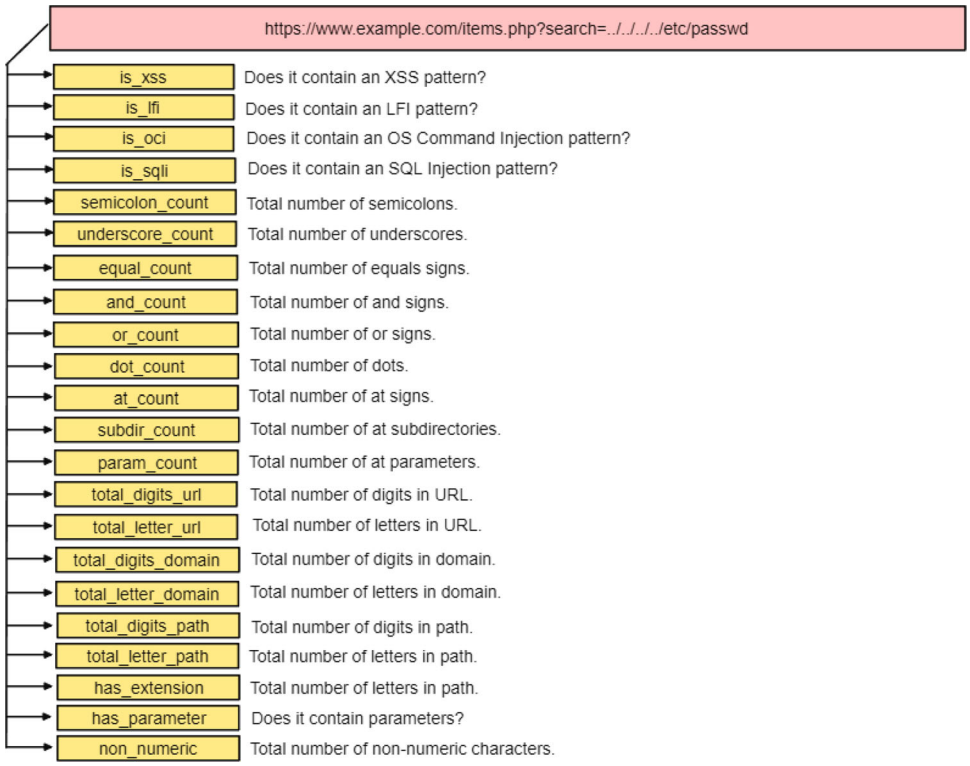

Fig. 8 Feature engineering for URL analysis: Comprehensive list of security-relevant features extracted from URLs, including their descriptions and relevance for malicious URL detection

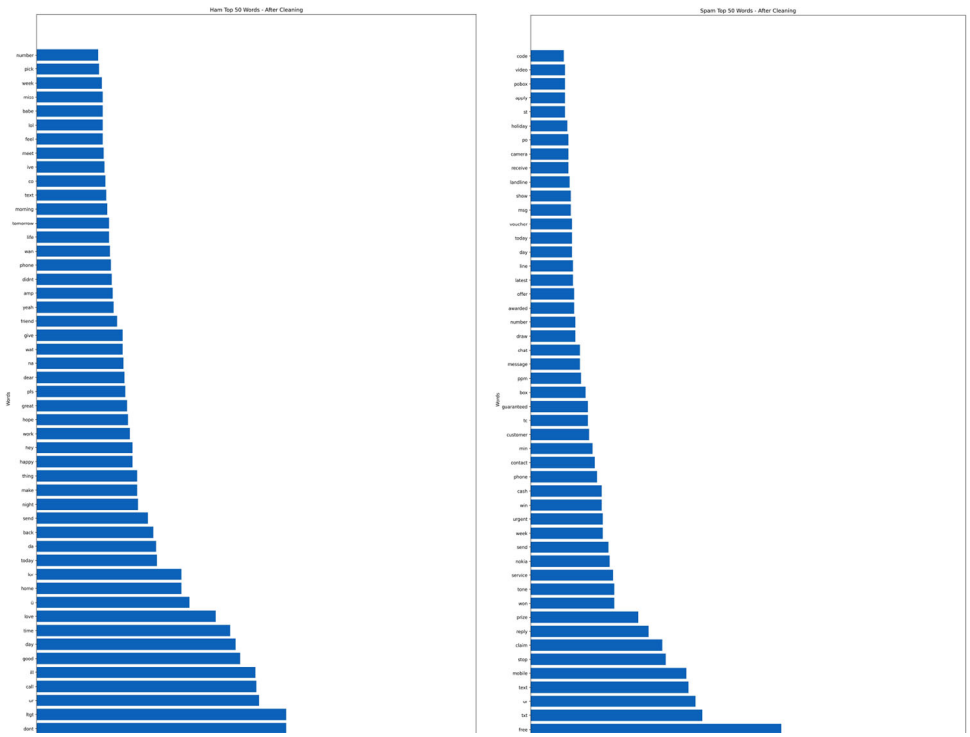

Fig. 9 Comparative word frequency analysis between ham and spam SMS messages after text preprocessing: (Left) Top 50 characteristic unigrams in legitimate messages (ham), (Right) Most frequent tokens in spam messages, revealing key discriminative terms (e.g., ‘free’, ‘voucher’, ‘code’). Bar heights represent normalized term frequencies across each class

Table 4 Optimized hyperparameter values for each deep learning architecture: Comparative configurations showing batch sizes, optimization algorithms, learning rates, and training epochs tailored to network intrusion (KDDCUP-99), URL detection (FWAF), and spam classification (SMSS-pam) tasks

Dataset Name	Model Name	Batch Size	Optimizer	Learning Rate	Epochs
IDS	CNN-LSTM	1024	RMSprop	0.01	10
	CNN	1024	RMSprop	0.01	6
	DNN	1024	Adam	0.001	8
	LSTM	1024	Adam	0.001	15
	GRU	1024	RMSprop	0.01	8
	RNN	1024	Adam	0.001	14
Query	CNN-LSTM	1024	RMSprop	0.001	10
	CNN	1024	RMSprop	0.001	10
	DNN	1024	RMSprop	0.001	10
	LSTM	1024	Adam	0.001	10
	GRU	1024	RMSprop	0.001	10
	RNN	1024	RMSprop	0.001	2
SMS Spam	CNN-LSTM	1024	RMSprop	0.001	18
	CNN	1024	Adam	0.01	6
	DNN	1024	Adam	0.001	5
	LSTM	1024	Adam	0.01	9
	GRU	1024	Adam	0.01	5
	RNN	1024	RMSprop	0.001	11

3.4 Evaluation of the Performance of DL Models

This study used Python language and "TensorFlow" library for model training. The models were run on a computer with 32 GB RAM (Random Access Memory) and 2x Tesla T4 16 GB graphics card. The models developed for all three data sets are given in Fig. 1. Since the dataset is unbalanced, Accuracy, Precision, Sensitivity, F1 Score, and ROC-AUC score were used as success metrics.

3.4.1 The Results of the IDS Dataset Model

Figure 10 shows the results of the models trained for IDS detection. The CNN-LSTM model showed the best performance with a test performance of 99.95%, precision performance of 99.92%, recall performance of 99.86%, and F1 score of 99.98%, and ROC-AUC score of 99.89% in Figure [12]. Figure [11] shows the confusion matrices of the methods. According to the confusion matrix of the CNN-LSTM model, which is the most successful DL model, only 49 out of 106.500 test data were misclassified. The CNN-LSTM model took 104.21 s for training and 14.67 s for testing.

Figure 10 shows that the CNN-LSTM model has the best training and validation performance during training compared to the other five models. According to Figs. 11 and 12, the CNN-LSTM model has the most successful test classification performance for the IDS dataset in studies with test data.

3.4.2 The Model Results for Query Data Set

The results of the models trained for malicious query detection are given in Fig. 13. The DNN model showed the best performance with 98.78% test performance, 84.50% precision performance, 91.45% recall performance, 87.83% F1 score, and 98.69% ROC-AUC score. The DNN model took 84.84 s for training and 12.19 s for testing.

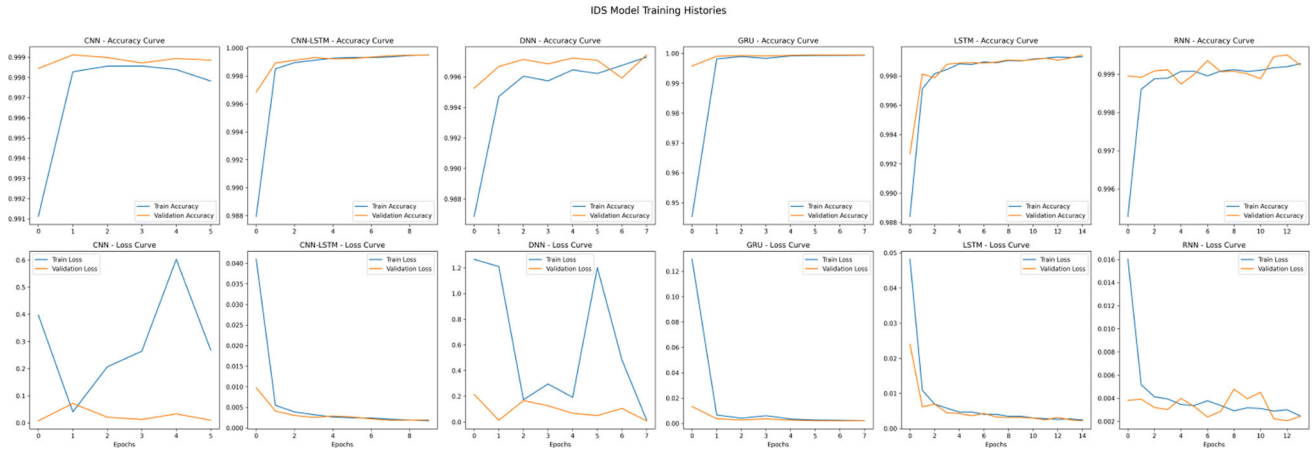

Fig. 10 Model training results on KDDCUP-99 dataset: Accuracy evolution (upper panel) and loss reduction (lower panel) during training/validation for compared architectures. Yellow lines indicate validation set performance

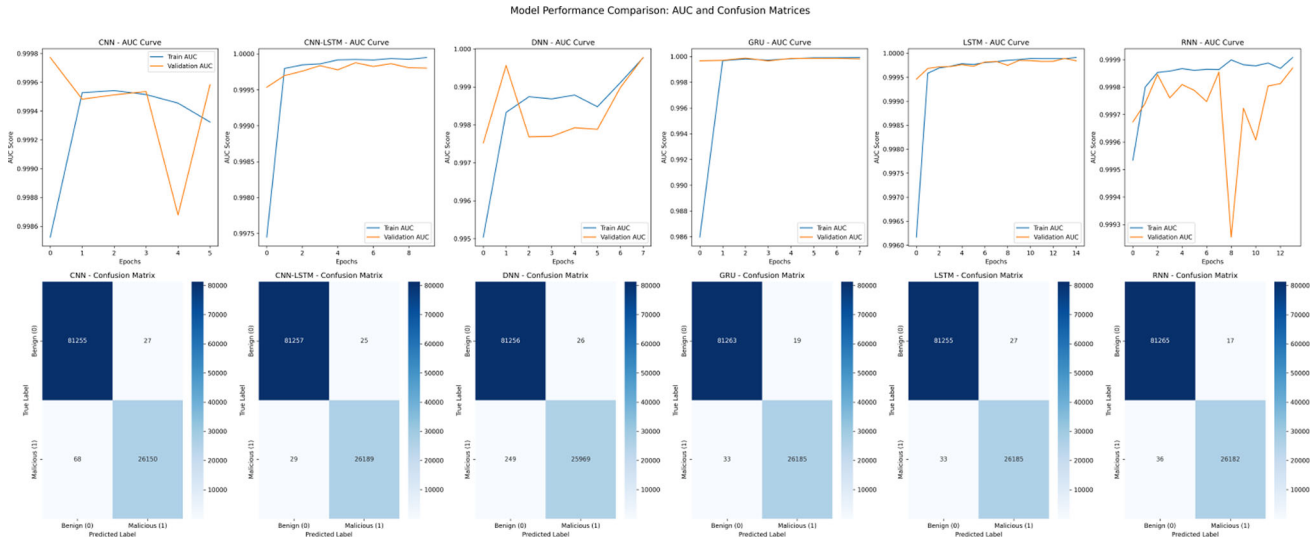

Fig. 11 Test results for intrusion detection systems: ROC analysis (top) and confusion matrices (bottom) comparing models on the complete KDDCUP-99 test suite

According to Figs. 14 and 15, DNN has the most successful test classification performance for the Query dataset in studies with test data.

3.4.3 The Model Results for the SMS-Spam Data Set

The results of the trained models for SMS-Spam detection are given in Fig. 16 The CNN-LSTM model showed the best performance with a test performance of 99.09%, precision performance of 98.05%, recall performance of 95.22%, F1 score of 96.59%, and ROC-AUC score of 99.45%. The CNN-LSTM model takes 12.71 s for Training and 1.61 s for Testing.

According to Fig. 17, the CNN-LSTM model has the best training and validation performance during training compared to the other five models.

According to Figs. 17 and 18, CNN-LSTM has the most successful test classification performance for the SMS-Spam dataset in studies with test data.

 https://doi.org/10.1007/s44196-025-00990-6

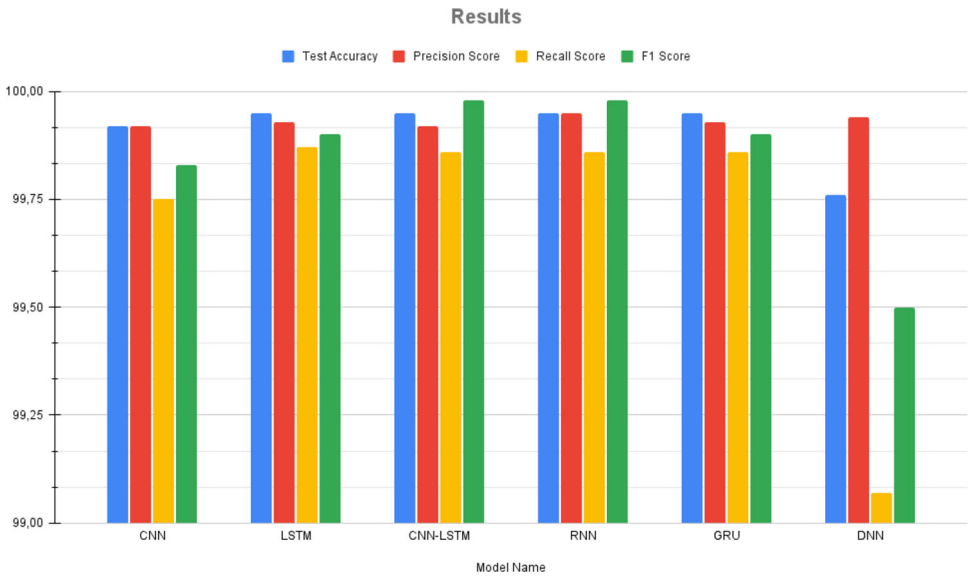

Fig. 12 Test performance comparison for IDS models: Multi-metric evaluation showing accuracy (blue), precision (red), recall (yellow), and F1 score (green) across different architectures. The CNN-LSTM model achieves optimal balance between detection rate (recall = 99.86) and false alarms (precision = 99.92)

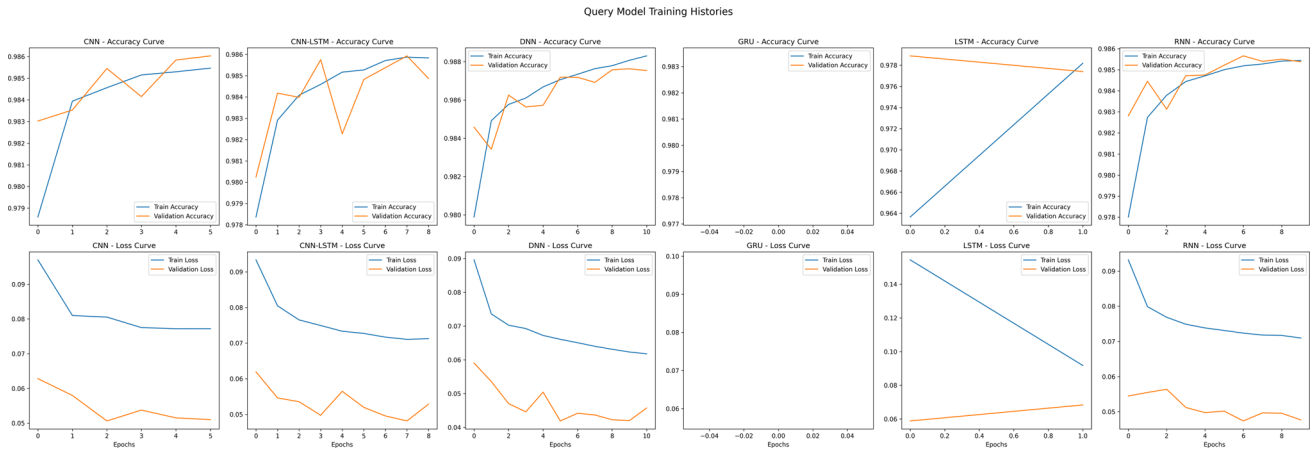

Fig. 13 Model training results on FWAF dataset: Accuracy evolution (upper panel) and loss reduction (lower panel) during training/validation for compared architectures. Yellow lines indicate validation set performance

3.4.4 Comparison of Machine Learning Methods Versus Deep Learning Methods Results

In this study, traditional machine learning (ML) methods were also applied as a comparison to enable a more comprehensive evaluation of the performance of the proposed deep learning (DL) models. While the primary focus is on deep learning architectures, the lightGBM, CatBoost, and XGBoost algorithms were also trained on the same datasets, and their F1 score results are presented in a comparative manner in Table 5

According to the F1 score comparisons presented in Table 5:

- **KDDCUP-99**: DL models (CNN-LSTM: 99.98) provided a 0.02% absolute performance increase compared to traditional ML methods (CatBoost: 99.96).
- **FWAF**: DL models (DNN: 87.83) achieved a 6.8% absolute performance increase compared to traditional ML methods (Ensemble: 82.24).
- **SMSpamCollection**: DL models (CNN-LSTM: 96.59) provided a 3.04% absolute performance increase compared to traditional ML methods (Ensemble: 93.54).

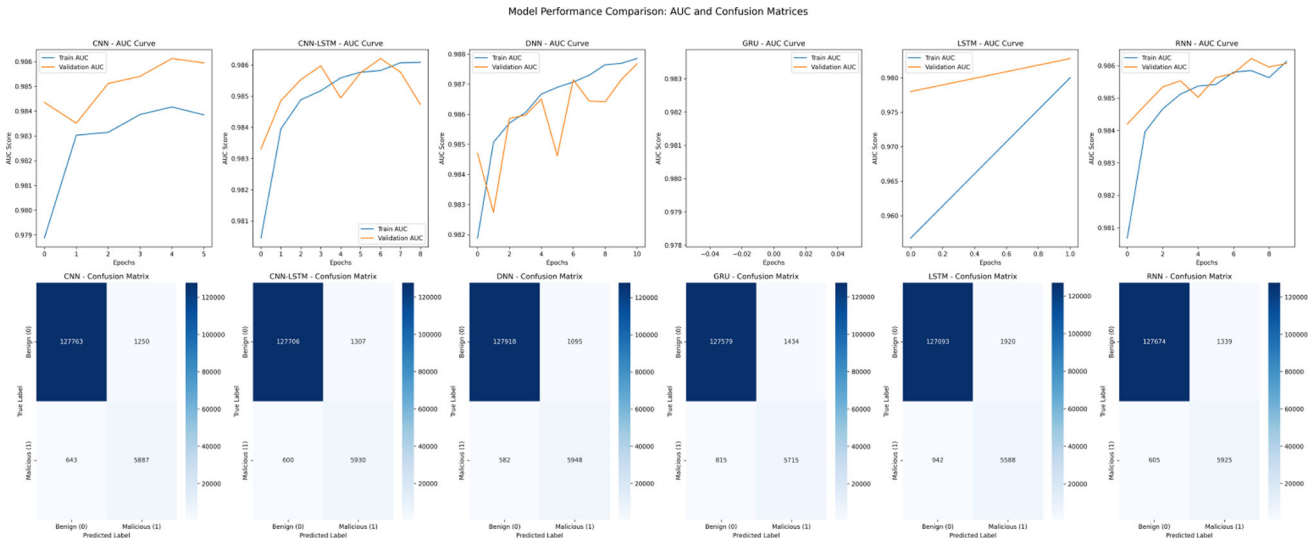

Fig. 14 Test results for malicious URL detection models: ROC analysis (top) and confusion matrices (bottom) comparing models on the complete FWAF test suite

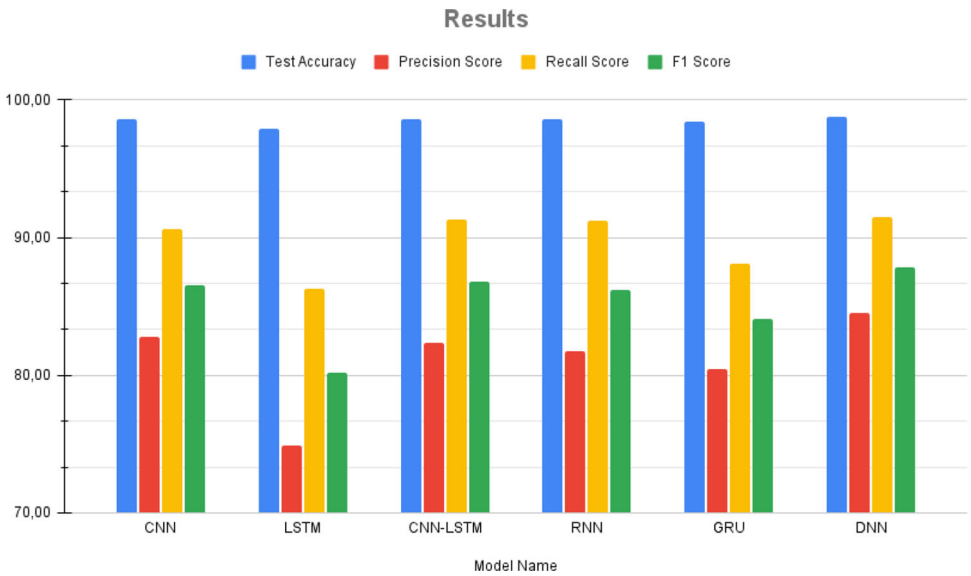

Fig. 15 Test performance comparison for malicious URL detection models: Multi-metric evaluation showing accuracy (blue), precision (red), recall (yellow), and F1 score (green) across different architectures. The DNN model achieves optimal balance between detection rate (recall=87.83) and false alarms (precision=91.45)

3.5 System Implementation

Wireshark listened to the network traffic for 5 s and saved it in a ".pcap" file. The feature was then extracted using this file. For this, the ".pcap" file was converted into a suitable fo mat. The "tcmdump" tool was used for this process. Tcpdump is a command line tool for capturing and analyzing network traffic. It was used to monitor, record, and analyze packets with various filters. With this tool, it filters the columns in the pcap file, filters out the unnecessary rows for the most successful DL model, and brings it into a format suitable for feature extract on. We used Python's "Scapy" library to extract features from the resulting filtered file. Scapy is the library used to manipulate network protocols with Python programming language to create, send, capture, and analyze packets on the network. By reading the packets from the filtered file with Scapy, features were extracted from these packets. Finally, these features were tabulated and sent to our most successful DL model to classify them according to the data set.

 https://doi.org/10.1007/s44196-025-00990-6

Fig. 16 Model training results on SMSSpamCollection dataset: Accuracy evolution (upper panel) and loss reduction (lower panel) during training/validation for compared architectures. Yellow lines indicate validation set performance

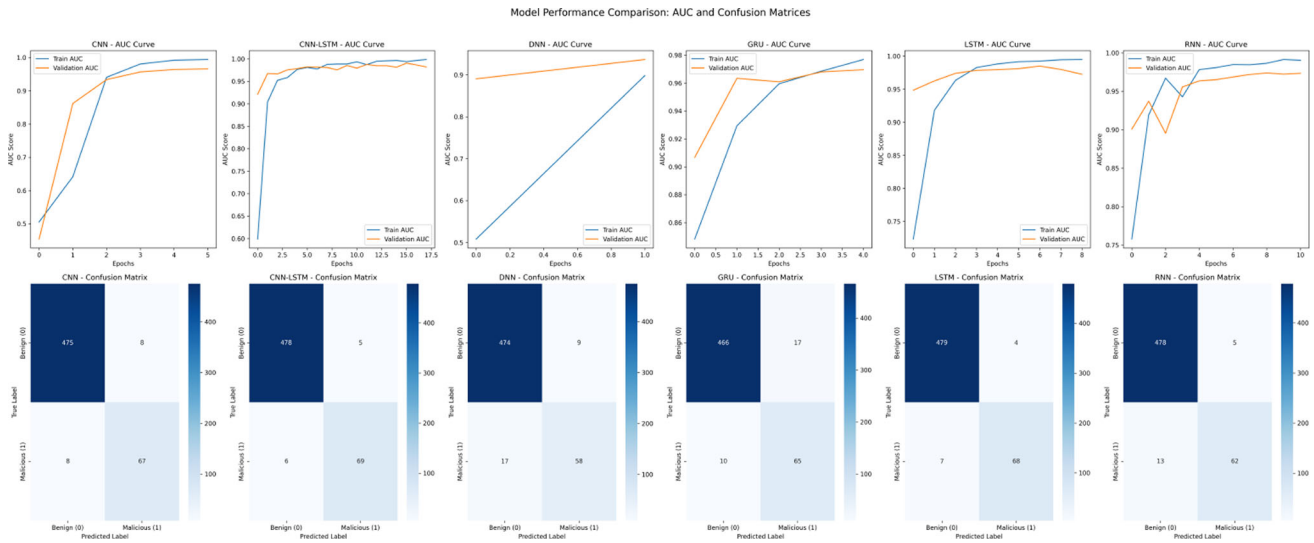

Fig. 17 Test results for spam detection models: ROC analysis (top) and confusion matrices (bottom) comparing models on the complete SMSSpamCollection test suite

The process steps to run the IDS detection model in real-time are given in Fig. 19. A browser plugin was written to test the malicious URL detection model in real-time. We capture every page the user enters and send it to the plugin. The plugin sends the address to the API to be classified by the model.

A browser plugin was also written to test the spam detection model in real-time. According to Fig. 20, the text given by the user as input was sent to the API with the help of the plugin, and the message was classified.

The 'AdminLTE' template followed the created models and monitored the results. AdminLTE is an open-source admin dashboard and control panel. According to Fig. 21, a web application was developed using this template with the Flask library.

3.6 Limitations

Although the deep learning models developed in this study for three different cybersecurity problems have yielded successful results, they contain various limitations in terms of problem level, data set specificity, and hardware

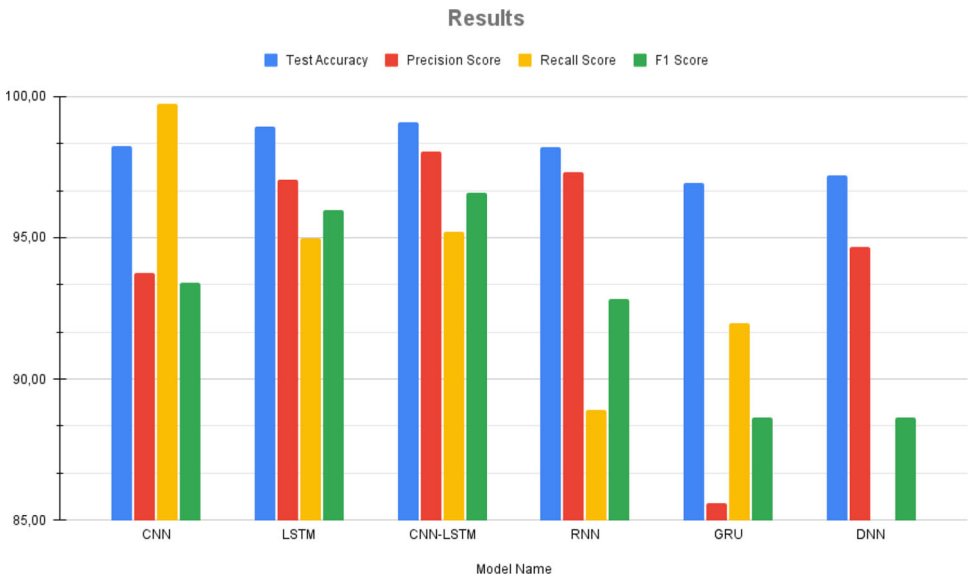

Fig. 18 Test performance comparison for Spam detection models: Multi-metric evaluation showing accuracy (blue), precision (red), recall (yellow), and F1 score (green) across different architectures. The CNN-LSTM model achieves optimal balance between detection rate (recall=95.22) and false alarms (precision=98.05)

Table 5 Model performance comparison across datasets (F1 scores)

Type	Model name	KDDCUP-99	FWAF	SMSSpam
DL	CNN	99.83	86.54	93.40
DL	LSTM	99.90	80.18	95.97
DL	CNN-LSTM	**99.98**	86.80	**96.59**
DL	RNN	99.98	86.23	92.84
DL	GRU	99.90	84.08	88.65
DL	DNN	99.50	**87.83**	88.65
ML	LightGBM	98.62	81.47	92.19
ML	CatBoost	**99.96**	81.33	90.64
ML	XGBoost	99.95	80.54	89.65
ML	Ensemble	99.95	**82.24**	**93.54**

Note: All values represent F1 scores on test sets. Best results in each column are highlighted in bold

conditions. Being aware of these limitations is important for evaluating the applicability of the system in the real world and reveals areas for improvement in future studies.

In terms of problem-based limitations, the three main problems addressed in this study (intrusion detection, malicious URL detection and spam message classification) are critical in today's cybersecurity world, but these problems are modeled in a limited scope. For each problem, a single model and a single classification strategy were used. In the real world, threat types are much more diverse and attack surfaces are more dynamic and broad. For example, intrusion detection systems may need to analyze not only network traffic but also different types of data such as system logs, file accesses, etc. Malicious URL detection is based on the URL text only, but features such as the page content, hosting service and geographic location are not included in the model. The lack of such enriched features may lead to the model failing to detect some advanced attack types. Future work is planned to address these problems with more comprehensive and multimodal data sources.

In terms of Data Set Based Limitations, the data sets used in this study (KDDCUP-99, FWAF and SMSSpam-Collection) are benchmark data sets that are frequently preferred in the literature and used in various academic comparisons. However, these data sets also have some limitations. For example, the KDDCUP-99 dataset contains traffic data from the 1990s and may be insufficient to represent the advanced and complex attack types encountered today. While the FWAF dataset contains real URL samples, it may be outdated in the face of constantly updated malware techniques. The SMSSpamCollection dataset is more of a collection of individual user messages and

 https://doi.org/10.1007/s44196-025-00990-6

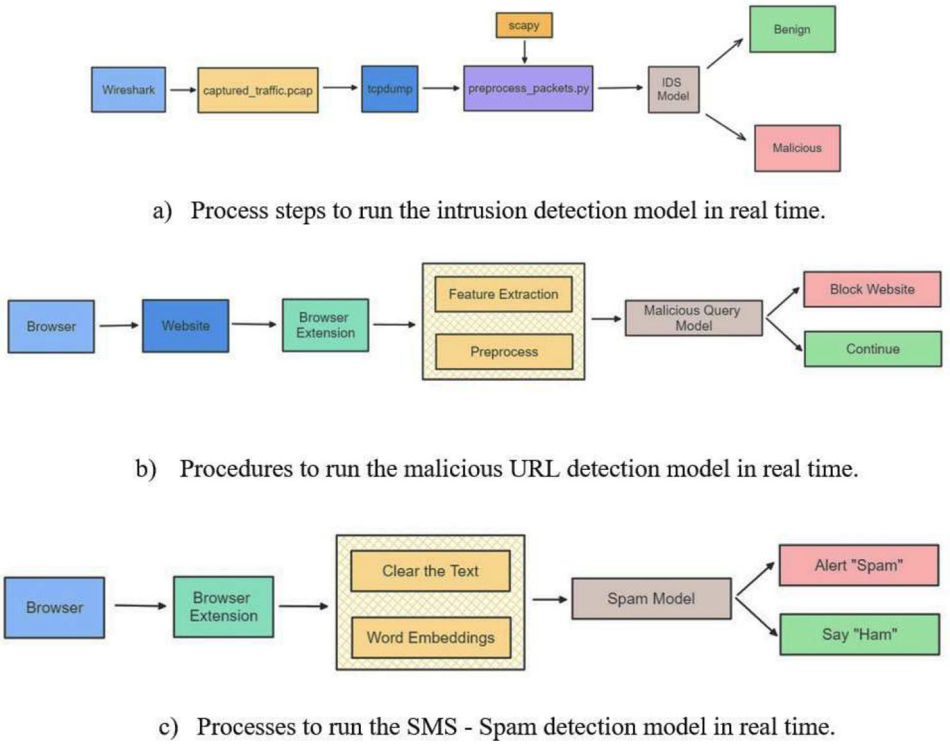

Fig. 19 End-to-end implementation of the multi-threat detection system: Real-time processing pipeline for (**a**) Network intrusion detection, (**b**) Malicious URL detection, and (**c**) Spam detection, featuring automated feature extraction and parallel model inference. Color-coded blocks represent distinct processing stages

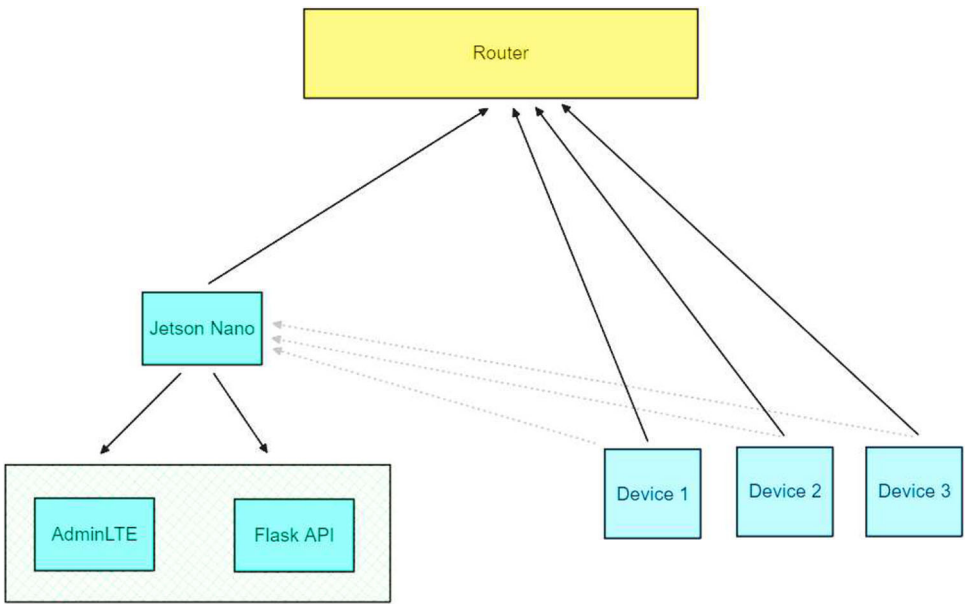

Fig. 20 The developed real-time firewall application

may not cover corporate spam scenarios. In the future, the capabilities of the model will be re-evaluated with more up-to-date and multidimensional data sets.

In terms of hardware limitations, a Jetson Nano device with 4 GB RAM was chosen to demonstrate the integration of the system developed in this study into an edge device. While such low-power hardware is important for real-time operability, it may be limited in dealing with high volumes of data. In particular, the inference processes of time series based networks such as LSTM and GRU are memory and processor resource intensive. Due to the limited processing power of the device, performance degradation may occur in the face of simultaneous multi-user traffic or high-speed incoming data streams. Such limitations should be taken into account when integrating the system

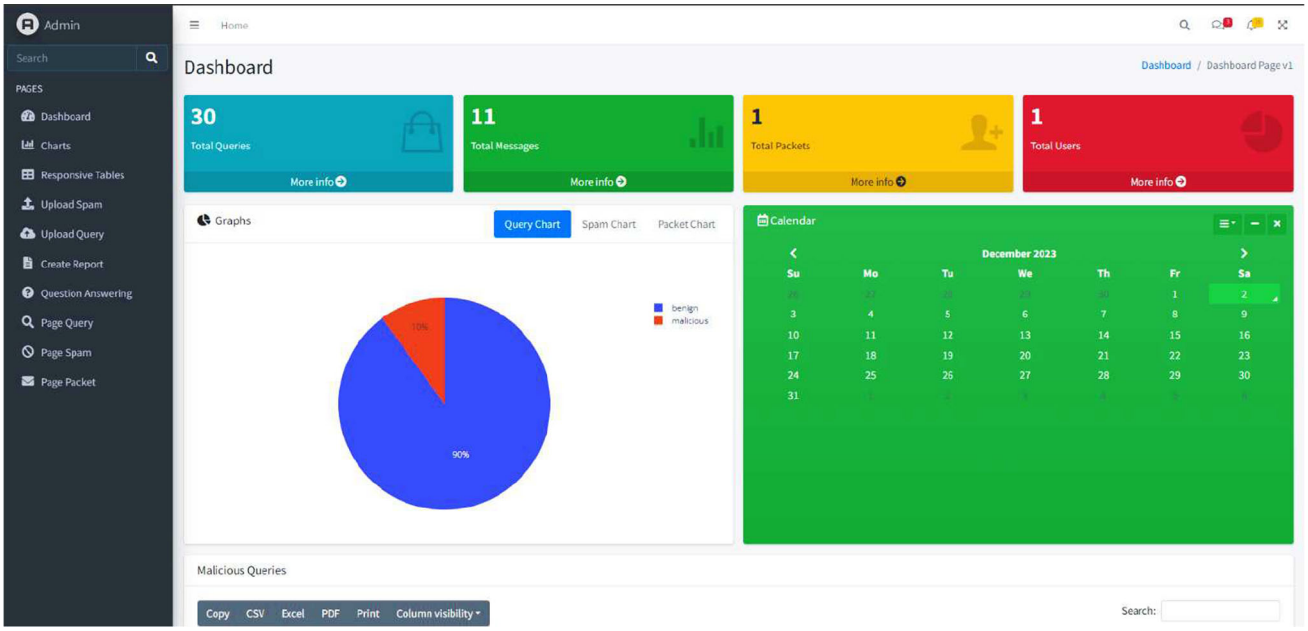

Fig. 21 The home page screen of the developed web application

into a production environment. Future work aims to overcome these limitations with more powerful hardware or model optimization methods.

4 Conclusion

Traditional methods used in cybersecurity today are unable to provide sufficient response time and flexibility against threats. The need to manually update rule-based firewall systems in dynamic environments where threats are constantly evolving and new types of attacks are emerging seriously limits the effectiveness of these systems. Therefore, the need for more proactive, learnable, and automated AI-based solutions is increasing day by day. Deep learning (DL)-based systems are considered an important alternative in the field of security due to their superior performance in identifying and classifying complex patterns. In this study, three fundamental problems in cybersecurity are addressed: Network Intrusion Detection (using the KDDCUP-99 dataset with a CNN-LSTM model, achieving a 99.98% F1 score), Malicious URL Detection (using the FWAF dataset with a DNN model, achieving an 87.73% F1 score), and Spam Detection (using the SMSSpamCollection dataset with a CNN-LSTM model, achieving a 96.69% F1 score). The deep learning models developed for each problem area were compared with various machine learning (ML)-based approaches, and it was found that DL models generally achieved higher accuracy rates. This once again highlights the importance of the representation learning advantage provided by DL methods, especially in areas where complex and unstructured data are used. In addition, the developed system has been tested not only theoretically but also practically. The prototype system developed on Jetson Nano hardware was able to capture real-time traffic through Wireshark and special browser extensions and classify this data instantly through trained DL models. Thus, the applicability of deep learning-based security solutions even in resource-constrained embedded systems has been demonstrated. In conclusion, this study clearly demonstrates the potential of deep learning in cybersecurity in terms of both theoretical success and practical applicability. In the future, the accuracy and processing efficiency of the system can be improved by using models trained with more up-to-date and large-scale data sets, testing different architectures (e.g., Transformer-based structures), and integrating model compression and acceleration techniques. Additionally, the development of multi-task models

that enable multi-dimensional analysis of threats will allow security systems to become more comprehensive and adaptive.

Acknowledgements This study was supported by both Istanbul Sabahattin Zaim University BAP-2023-36 and TUBITAK 2209 - 1919B012322054.

Author Contributions Bayrak and Karaca contributed to the design of the algorithms and the data acquisition. Toson, Tayfur, and Yavas wrote some sections, and performed the final corrections. All authors have read and agreed to the published version of the manuscript.

Funding The authors have not received any financial support for this research.

Data Availability No datasets were generated or analyzed during the current study.

Declarations

Conflict of Interest The authors declare no conflict of interest.

Ethical Approval Ethical approval was not required for this paper.

References

1. Aktar, S., Nur, A.Y.: Towards ddos attack detection using deep learning approach. Comput. Secur. **129**, 103251 (2023). https://doi.org/10.1016/j.cose.2023.103251
2. Salmi, S., Oughdir, L.: Performance evaluation of deep learning techniques for dos attacks detection in wireless sensor network. J. Big Data **10**(1), 17 (2023). https://doi.org/10.1186/s40537-023-00692-w
3. Wang, W., Du, X., Shan, D., Qin, R., Wang, N.: Cloud intrusion detection method based on stacked contractive auto-encoder and support vector machine. IEEE Trans. Cloud Comput. **10**(3), 1634–1646 (2020)
4. Kim, J., Kim, J., Kim, H., Shim, M., Choi, E.: Cnn-based network intrusion detection against denial-of-service attacks. Electronics **9**(6), 916 (2020)
5. Ghanem, R., Erbay, H.: Spam detection on social networks using deep contextualized word representation. Multimedia Tools Appl. **82**(3), 3697–3712 (2023). https://doi.org/10.1007/s11042-022-13397-8
6. Altuncu, M.A.: "Makine öğrenmesi ve derin öğrenme yöntemleri kullanılarak saldırı tespit ve önleme sistemi geliştirilmesi". Kocaeli Üniversitesi (2021)
7. Demir, S., Aslan, Z.: K-nn, nn ve feature selection yöntemleri ile firewall verilerinin sınıflandırması. Anadolu Bil Meslek Yüksekokulu Dergisi **17**(66), 139–148 (2023)
8. Özekes, S., Karakoç, E.N.: Makine öğrenmesi yöntemleriyle anormal ağ trafiğinin tespit edilmesi. Düzce Üniversitesi Bilim ve Teknoloji Dergisi **7**(1), 566–576 (2019)
9. Demir, F.: Siber saldırı tespiti için makine öğrenmesi yöntemlerinin performanslarının incelenmesi. Balıkesir Üniversitesi Fen Bilimleri Enstitüsü Dergisi **23**(2), 782–791 (2021)
10. Sahingoz, O.K., Çebi, C.B., Bulut, F.S., Fırat, H., Karataş, G.: Saldırı tespit sistemlerinde makine öğrenmesi modellerinin karşılaştırılması. Erzincan Univ. J. Sci. Technol. **12**(3), 1513–1525 (2019)
11. Seyyar, Y.E., Yavuz, A.G., Ünver, H.M.: Detection of web attacks using the BERT model. In: 2022 30th Signal Processing and Communications Applications Conference (SIU), 1–4 (2022)
12. Ilhan, K.: WEB trafik verilerinde yapay bağışıklık algoritmaları ile anomali tespiti (Master's thesis, Bilecik Şeyh Edebali Üniversitesi, Fen Bilimleri Enstitüsü) (2019)
13. Gul, A.: Ağ davranış modeli ile kurum içi saldırıların belirlenmesi (Master's thesis, Bilişim Enstitüsü) (2018)

14. Ahmetoglu, H., Daş, R.: Makine öğrenmesi yöntemleri kullanarak web uygulama saldırılarının tespitinde genetik öznitelik seçimi yaklaşımı. Türkiye Bilişim Vakfı Bilgisayar Bilimleri ve Mühendisliği Dergisi **14**(2), 109–119 (2021)
15. Yu, L., Dong, J., Chen, L., Li, M., Xu, B., Li, Z., Qiao, L., Liu, L., Zhao, B., Zhang, C.: Pbcnn: packet bytes-based convolutional neural network for network intrusion detection. Comput. Netw. **194**, 108117 (2021). https://doi.org/10.1016/j.comnet.2021.108117
16. Brown, A., Gupta, M., Abdelsalam, M.: Automated machine learning for deep learning based malware detection. Comput. Secur. **137**, 103582 (2024). https://doi.org/10.1016/j.cose.2023.103582
17. Paul, A., Sharma, V., Olukoya, O.: Sql injection attack: detection, prioritization & prevention. J. Inf. Secur. Appl. **85**, 103871 (2024). https://doi.org/10.1016/j.jisa.2024.103871
18. Dawadi, B.R., Adhikari, B., Srivastava, D.K.: Deep learning technique-enabled web application firewall for the detection of web attacks. Sensors **23**(4), 2073 (2023). https://doi.org/10.3390/s23042073
19. Koksal, K., Dogan, B., Altikardes, Z.A.: Topluluk sınıflandırıcı kullanarak zararlı url tespiti. Veri Bilimi **4**(3), 113–122 (2021)
20. Ucar, E., Ucar, M., İncetaş, M.O.: A Deep learning approach for detection of malicious URLs. In: 6th International Management Information Systems Conference (pp. 12–20) (2019)
21. Abdi, F.D., Wenjuan, L.: Malicious url detection using convolutional neural network. J. Int. J. Comput. Sci. Eng. Inf. Technol. **7**(6), 1–8 (2017)
22. Tiryaki, F., Şentürk, Ü., Yücedağ, İ: Developing and evaluating an artificial intelligence model for malicious url detection. Avrupa Bilim ve Teknoloji Dergisi **47**, 13–17 (2023)
23. Moghimi, M., Varjani, A.Y.: New rule-based phishing detection method. Expert Syst. Appl. **53**, 231–242 (2016)
24. Ismail, M., Alrabaee, S., Choo, K.K.R., Ali, L., Harous, S.: A Comprehensive Evaluation of Machine Learning Algorithms for Web Application Attack Detection with Knowledge Graph Integration. Mobile Networks and Applications, 1-30 (2024)
25. Rao, S., Verma, A.K., Bhatia, T.: Hybrid ensemble framework with self-attention mechanism for social spam detection on imbalanced data. Expert Syst. Appl. **217**, 119594 (2023)
26. Lai, C.M., Chen, M.H., Kristiani, E., Verma, V.K., Yang, C.T.: Fake news classification based on content level features. Appl. Sci. **12**(3), 1116 (2022)
27. Zavrak, S., Yilmaz, S.: Email spam detection using hierarchical attention hybrid deep learning method. Expert Syst. Appl. **233**, 120977 (2023)
28. Ramya, S.P., Eswari, R.: A regularization based simple shallow perceptron network for detection of fake news in social networks. Multimedia Tools and Applications **1–21**,(2024). https://doi.org/10.1007/s11042-024-18320-x
29. Karamollaoglu, H.: Metin verilerinde içerik tabanlı spam tespiti (Master's thesis, Bilişim Enstitüsü)
30. Tekerek, A.: Support vector machine based spam sms detection. Politeknik Dergisi **22**(3), 779–784 (2019)
31. https://github.com/faizann24/Fwaf-Machine-Learning-driven-Web-Application-Firewall
32. https://www.kaggle.com/datasets/toobajamal/kdd99-dataset
33. https://archive.ics.uci.edu/dataset/228/sms+spam+collection
34. Wan, Q., Xu, X., Han, J.: A dimensionality reduction method for large-scale group decision-making using tf-idf feature similarity and information loss entropy. Appl. Soft Comput. **150**, 111039 (2024)
35. Johnson, S.J., Murty, M.R., Navakanth, I.: A detailed review on word embedding techniques with emphasis on word2vec. Multimedia Tools and Applications **83**(13), 37979–38007 (2024)
36. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543) (2014, October)
37. Shiri, F.M., Perumal, T., Mustapha, N., Mohamed, R.: A comprehensive overview and comparative analysis on deep learning models: CNN, RNN, LSTM, GRU. arXiv preprint arXiv:2305.17473 (2023)
38. Erçin, M.S., Yolaçan, E.N.: Web uygulamalarında enjeksiyon saldırılarının tespiti. Eskişehir Türk Dünyası Uygulama ve Araştırma Merkezi Bilişim Dergisi **5**(1), 1–11 (2024)
39. Pandi, S.S., Farook, A.M., Kingston, W.: Scenario based Deep Learning Prediction for Legal Judgment using LSTM and CNN. In 2023 Third International Conference on Smart Technologies, Communication and Robotics (STCR) (Vol. 1, pp. 1-6). IEEE (2023, December)
40. Daraghmi, E.Y., Qadan, S., Daraghmi, Y., Yussuf, R., Cheikhrouhou, O., Baz, M.: From Text to Insight: An Integrated CNN-BiLSTM-GRU Model for Arabic Cyberbullying Detection. IEEE Access (2024)
41. Someswara Rao, C., Raminaidu, C., Raju, K.B., Sujatha, B.: Effective Fake News Classification Based on Lightweight RNN with NLP. Annals of Data Science, 1-25 (2024)
42. Sze, V., Chen, Y.H., Yang, T.J., Emer, J.S.: Efficient processing of deep neural networks: a tutorial and survey. Proc. IEEE **105**(12), 2295–2329 (2017)
43. Arslan, R.S.: Kötücül url filtreleme için derin öğrenme modeli tasarımı. Avrupa Bilim ve Teknoloji Dergisi **29**, 122–128 (2021)
44. Hossin, M., Sulaiman, M.N.: A review on evaluation metrics for data classification evaluations. International journal of data mining & knowledge management process **5**(2), 1 (2015)
45. Lamping, U., Warnicke, E.: Wireshark user's guide. Interface **4**(6), 1 (2004)

46. Manolescu, D., Reid, D., Secco, E.L.: Hardware and Software Integration of Machine Learning Vision System Based on NVIDIA Jetson Nano. In Future of Information and Communication Conference (pp. 129-137). Cham: Springer Nature Switzerland (2024, March)

Authors and Affiliations

Sengul Bayrak[1] · Alper Karaca[1] · Ferhat Toson[1] · Mehmet Emin Tayfur[1] · Selcuk Yavas[1]

✉ Sengul Bayrak
bayraksengul@ieee.org

Alper Karaca
alper.krc@outlook.com

Ferhat Toson
ferhattoson@hotmail.com

Mehmet Emin Tayfur
mehmetemin.tayfur@outlook.com

Selcuk Yavas
selcuk.yavas@outlook.com

[1] Department of Computer Engineering, Istanbul Sabahattin Zaim University, Halkali Street, 34303 İstanbul, Türkiye