Hindawi
Security and Communication Networks
Volume 2023, Article ID 2415288, 17 pages
https://doi.org/10.1155/2023/2415288

Research Article

MC-MLDCNN: Multichannel Multilayer Dilated Convolutional Neural Networks for Web Attack Detection

Nazanin Moarref and Mehmet Tahir Sandıkkaya

Department of Computer Engineering, Istanbul Technical University, Sarıyer, Istanbul, TR 34469, Türkiye

Correspondence should be addressed to Nazanin Moarref; moarref@itu.edu.tr

Received 10 March 2023; Revised 11 December 2023; Accepted 13 December 2023; Published 22 December 2023

Academic Editor: De Rosal Ignatius Moses Setiadi

The explosive growth of web-based technology has led to an increase in sophisticated and complex attacks that target web applications. To protect against this growing threat, a reliable web attack detection methodology is essential. This research aims to provide a method that can detect web attacks accurately. A character-level multichannel multilayer dilated convolutional neural network (MC-MLDCNN) is proposed to identify web attacks accurately. The model receives the full text of HTTP requests as inputs. Character-level embedding is applied to embed HTTP requests to the model. Therefore, feature extraction is carried out automatically by the model, and no additional effort is required. This approach significantly simplifies the preprocessing phase. The methodology consists of multichannel dilated convolutional neural network blocks with various kernel sizes. Each channel involves several layers with exponentially increasing dilation sizes. Through the integration of multichannel and multilayer dilated convolutional neural networks, the model can efficiently capture the temporal relation and dependence of character granularity of HTTP requests at different scales and levels. As a result, the structure enables the model to easily capture dependencies over extended and long sequences of HTTP requests and consequently identify attacks accurately. The outcomes of the experiments carried out on the CSIC 2010 dataset show that the proposed model outperforms several state-of-the-art deep learning-based models in the literature and some traditional deep learning models by identifying web attacks with a precision score of 99.65%, a recall score of 98.80%, an F_1 score of 99.22%, and an accuracy score of 99.36%. A useful web attack detection system must be able to balance accurate attack identification with minimizing false positives (identifying normal requests as attacks). The success of the model in recognizing normal requests is further evaluated to guarantee increased security without sacrificing web applications' usability and availability.

1. Introduction

1.1. Background. Web applications are the gate to a great deal of sensitive data. The convenience of the Internet enables a large number of attackers to interact with web applications. Attackers have been able to conduct massive attacks more swiftly due to sophisticated and well-planned attack strategies through a variety of network technology tools. Web attacks continue to grow in both frequency and severity daily. Internet users are particularly at risk for threats that could result in monetary loss, identity theft, data fraud, and a loss of trust in conducting business. It is estimated that by 2025, the financial loss will amount to $10.5 trillion [1]. Researchers are developing new theoretical advancements to improve web security to reduce web attacks [2].

1.2. Limits of Prior ATRs. For more than a decade, numerous studies have provided solutions that make use of machine learning (ML) approaches to address web attack detection difficulties. However, it is believed that these kinds of solutions have only achieved a limited level of adoption in practice since they demand a significant amount of expert effort to develop and maintain [3]. For many solutions in this discipline, feature engineering is necessary to attain the desired performance. The current approaches focus on using bag-of-words feature engineering techniques, which reveal details about the entity of a word included inside the dataset. As a result, these techniques suffer from a lack of precise representation of the sequence in the dataset. Moreover, ML feature representations must be updated to keep up with the most recent web attacks. Given that feature engineering is frequently considered to be the most time-consuming component of developing an ML system, deep learning has drawn considerable attention for the detection of web attacks.

Deep learning algorithms are a branch of ML that employs artificial neural networks. They are better than conventional ML techniques as they can automatically learn the expressive feature representation, the lexical pattern, and the sequencing pattern of a given input while generalizing the expressive data representation. They have demonstrated an outstanding performance in many research fields such as image processing [4], natural language processing [5], speech recognition [6], computer vision [7], human activity recognition [8–12], and cyber security fields [13, 14]. In recent years, 1D convolutional neural networks (1D CNNs) in particular have shown an impressive performance for text classification [15, 16]. A considerable amount of research has also been conducted for web security based on deep learning-based approaches [17, 18]. The preprocessing stage of HTTP requests is greatly simplified at the character level in a number of deep learning-based approaches in the literature [2, 19]. These methods considered the HTTP requests as a series of characters and attempted to capture the character granularity dependencies to extract the pattern of attacks.

Despite the state-of-the-art performance of deep learning applications for web attack detection, they still have limitations when it comes to capturing long-term relations. For instance, in order to cover longer sequences, convolutional neural networks (CNNs) require larger receptive fields. Wider receptive fields necessitate more layers, and additional layers lead to additional parameters and a more challenging training procedure [20]. Another effective technique for handling sequential data is the use of long short-term memory networks (LSTMs), in which the receptive field may equal the entire input. Yet, due to their challenges in handling the issue of vanishing/exploding gradients, LSTMs still have difficulties in learning very long-term relations [21].

1.3. Research Motivation. Dilated convolutional neural networks (DCNNs) can be considered as a middle point of CNNs and LSTMs [22]. Dilated CNN is a type of CNN in which the filter has a defined spacing or dilation rate, enabling the network to increase its receptive field without adding more parameters. Dilated CNNs have the ability to significantly grow receptive fields without affecting resolution or coverage [23]. Dilated CNNs have lately achieved remarkable success in image segmentation [24], text classification, and text-to-speech [25, 26]. This study's main goal is to accurately identify web attacks using the full text of HTTP requests while keeping the preprocessing phase as simple as possible. Hence, the HTTP requests are regarded as sequences of characters. Dilated CNNs' effectiveness in capturing long-term relations and dependencies in comparison to the shortcomings of LSTMs and CNNs served as our motivation. In this study, a model based on dilated CNNs is developed as a multichannel multilayer dilated CNNs (MC-MLDCNNs). The model takes the full text of HTTP requests as the input, prepossesses them at the character level, and passes them as vectors to the model for the detection task.

The proposed methodology is developed exclusively for the detection of complicated and challenging patterns of web attacks that traditional security techniques could fail to recognize. It is expected that the proposed model will perform better than several existing models in terms of accuracy, false positive rate (FPR), precision, recall, and F_1 score. It is predicted that the proposed methodology will show its efficiency in identifying web attacks, hence improving the security posture of online systems and services.

1.4. Main Contribution. This study proposes a character-level multichannel multilayer dilated convolutional neural network (MC-MLDCNN). The model's architecture has the following benefits:

(i) Since the full text of HTTP requests is analyzed at the character level to automatically extract the relevant and important features, the preprocessing and feature extraction process is thereby greatly simplified. The character-level strategy makes the model adaptable and simple to apply since it does not rely on external efforts to derive key features of attacks.

(ii) The dilated CNNs in the model's structure help to address the shortcomings of LSTM and CNNs in terms of capturing long-term dependencies.

(iii) The model benefits from the use of many channels with different kernel sizes in order to extract a variety of temporal relationships from the requests.

(iv) Each channel consists of stacking layers of dilated CNNs with exponentially increasing dilation sizes. Accordingly, receptive fields are expanded exponentially. This strategy enables the model to extract various temporal relations at different levels, capture high-ordered feature interactions, and consequently discover complex and long-term dependencies of inputs.

To the best of our knowledge, character-level MC-MLDCNN is first specifically tailored to detect web attacks. The study's main contribution is the methodology's specifically designed structure, which is based on dilated CNNs. The effectiveness of the proposed methodology for identifying web attacks is investigated in the section Results and Discussion. According to the obtained results conducted on the CSIC 2010 dataset [27], the proposed methodology outperforms several competitive state-of-the-art models in the literature in terms of accuracy, false positive rate, recall, precision, and F_1 score. In the field of cyber security, our developed model for detecting web attacks represents a significant advancement that contributes to scientific knowledge in several ways.

1.4.1. Enhanced Attack Detection Accuracy. The proposed methodology closes a significant gap in current web attack detection solutions by concentrating on both short-term and long-term relationships of characters. Consequently, it can extract the complicated pattern of web attacks efficiently. In comparison to conventional methods, it achieves higher accuracy in identifying complex attacks by utilizing multichannel multilayer dilated CNNs. This increase in accuracy makes the defense against web-based attacks more reliable.

1.4.2. Reduced False Positives. Any successful cyber security system must focus on reducing false positive (identifying normal requests as attacks) alerts. The load on security workers will be reduced as a result of our model's reduction in false alarms, which also lowers the possibility that actual attacks might be missed. This enhancement helps create a more effective security infrastructure and improved resource allocation.

1.4.3. Protection against Evolving Attacks. Intricate patterns and hidden malicious activity are frequent components of difficult-to-recognize attacks over the Internet, which can elude detection by traditional security technologies. The flexibility and capability of the proposed methodology to identify evolving attacks assist in the ongoing scientific study of risks associated with cyber security.

1.4.4. Data-Driven Insights. The proposed methodology produces useful information and insights about new attack trends and patterns as part of its operation. To develop proactive security measures and advance the field of attack intelligence more broadly, such information can be studied to better comprehend cyberattacks.

1.4.5. Framework for Further Research. Future studies in the area of web attack detection can build on the basis provided by the proposed methodology. Researchers can expand on its architecture by adding new features and optimizing algorithms, promoting ongoing developments in the cyber security industry.

In a nutshell, the proposed methodology contributes to scientific knowledge by pushing the limits of what is possible in cyber security in addition to providing practical solutions to the critical challenge of complicated web attacks. It is a valuable advancement to the ongoing efforts for protecting modern systems and infrastructure against novel attacks due to its accuracy, decreased false positives, adaptability, and possibility for further research.

The proposed methodology could be deployed in many locations in a security architecture as shown in Figure 1. Figure 1(a) shows that MC-MLDCNN is located in between a firewall and a web server. Figure 1(b) shows that MC-MLDCNN is located in parallel to the web server to alert security operators. Figure 1(c) shows that MC-MLDCNN is used in place of a WAF, or even better, together with a WAF to enhance its efficacy.

2. Related Work

Mehta et al. carried out a comparative assessment of machine learning methods for the goal of detecting SQL injection, including logistic regression, random forest, SVM, naive Bayes, decision trees, gradient boost, K-means clustering, and KNN [28]. The findings of the experiment indicate that logistic regression performs best. Louk and Tama integrated bagging with gradient boosting decision tree (GBDT) techniques such as gradient boosting machine (GBM), LightGBM, CatBoost, and XGBoost to identify anomalies in an intrusion detection system [29]. According to the results, a combination of bagging and a gradient boosting machine (GBM) achieves the highest performance.

Deep learning methodologies have been progressively adopted in recent years and have shown promising results when compared to traditional ML approaches. The effectiveness of deep learning approaches over conventional ML techniques for the intrusion detection task is demonstrated in an experiment conducted by Althubiti et al. using the CSIC 2010 dataset [30]. Althubiti et al. extracted five important features to train the LSTM. The results of the study demonstrate that Althubiti et al.'s deep learning-based strategy beats a study [31] that used the same data with 9 extracted features and traditional ML techniques. Recurrent neural networks (RNNs) are also employed in a publication [32] to perform the same detection task on the NSL-KDD dataset. RNN is fed and trained after a feature engineering process. The model's performance is contrasted to benchmark ML algorithms. RNN performs better than ML models according to the results of the experiment. Fang et al. integrated LSTM with a bidirectional recurrent neural network (BRNN) to estimate the cyberattack rates [33]. Xing et al. conducted the experiments using self-collected data and compared the findings to hybrid models that included ML algorithms as well as statistical prediction models like ARIMA.

In a study for the identification of phishing attacks, Kasim makes use of both machine learning and deep learning techniques [34]. The features are encoded using a sparse autoencoder and a principal component in the proposed approach's feature engineering phase. With the aid of the light gradient boosted machine model (LightGBM), the encoded features are selected and categorized. The ISCX-URL dataset, which contains 77 distinct features, is used for evaluation. These features include measures related to the URL, host, domain, directory, file, and more. Out of the 154 features produced by the outputs of the principal component and autoencoder, the top 20 features for the study are chosen. In order to distinguish between regular HTTP and attack, Dawadi et al. conducted a comparison analysis between the two types of HTTP requests to extract attack-indicating characteristics and features using IDS

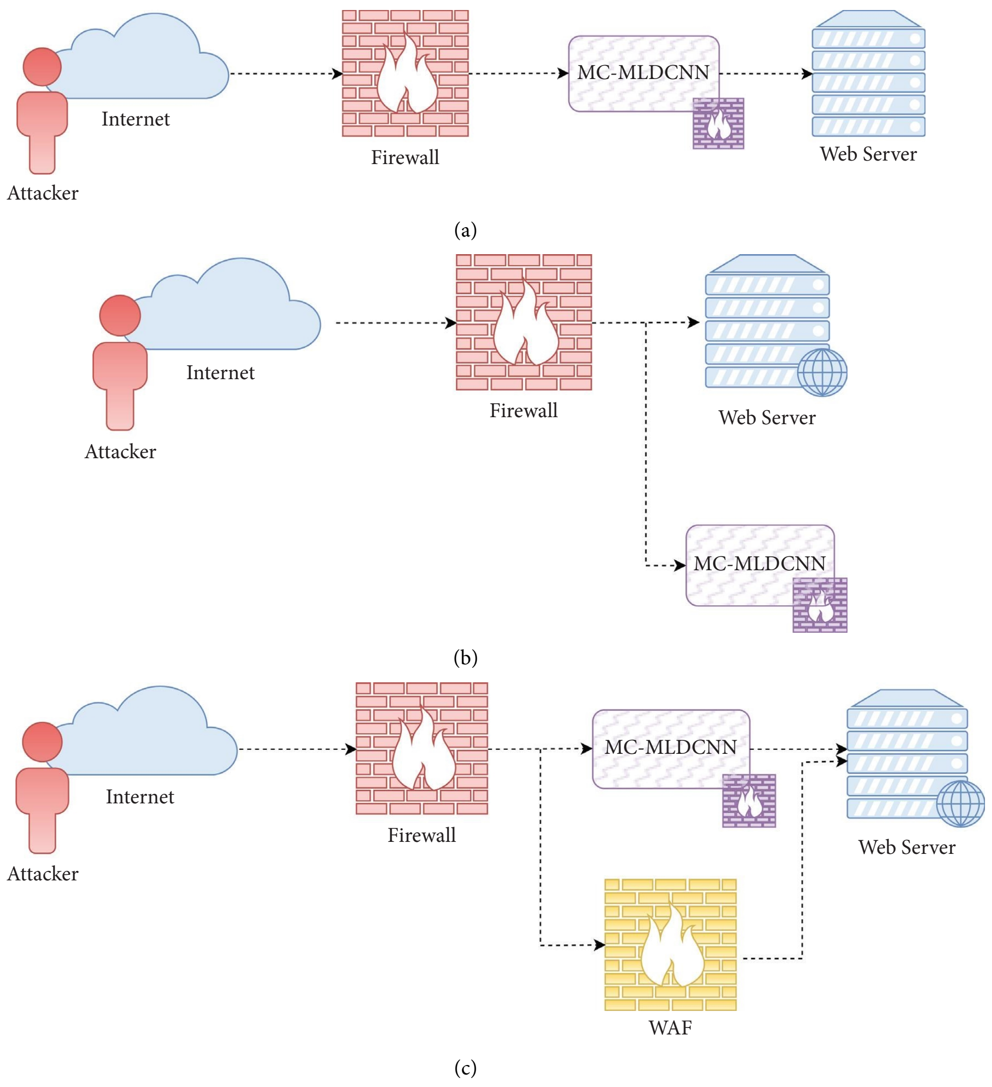

Figure 1: Examples of how the proposed model can be incorporated into real-world security architectures. (a) MC-MLDCNN is located in between a firewall and a web server. (b) MC-MLDCNN is located in parallel to the web server to alert security operators. (c) MC-MLDCNN is located in parallel to a WAF to enhance the efficacy.

ISCX 2012, 2019 DDoS CIC, and CISC 2010 datasets [35]. The relative features are given into a layered LSTM model for the task of attack detection after the feature engineering and preprocessing phase. For the evaluation stage, self-collected data are utilized. Although the discussed proposed models have achieved good performance, they are all highly reliant on feature engineering.

Hao et al. suggested a model based on stacked Bi-LSTM to detect web attacks [36]. The URL and the body of the post requests of the CSIC 2010 dataset are simply used. The word embedding technique is utilized to feed the input to the model. Each word is converted into a word vector using the Word2vec technique [37]. Similarly, Alaoui and Nfaoui used Word2vec to feed the CSIC 2010 dataset's HTTP method, HTTP request, and payload to the model [38]. Alaoui and Nfaoui employed an ensemble of LSTMs to identify web attacks. Zhang et al. suggested a word-level CNN utilizing the full text of the HTTP CSIC 2010 dataset [39]. Kernels of different sizes are used to convolve in the convolution layer of the model. A max-pooling layer is next applied to the outputs, and the results are passed to a fully connected layer for classification purpose.

Through the use of deep learning models, Tian et al. proposed a distributed system for web attack detection [40]. The methodology can be used in an Edge-of-Things (EoT) environment. FastText [41] and M-ResNet, a particular variation of ResNet [42], are incorporated in the proposed method. The URLs of the requests are converted to vectors using Word2vec and TF-IDF. After that, the vectors are concatenated and fed to the model. Similarly, Luo et al. developed an ensemble-based methodology to identify web attacks in a distributed environment [43]. The detection of web attacks is carried out individually using three deep learning models: M-ResNet, LSTM, and CNN. Using the results gathered from the models, an ensemble classifier then makes its final prediction. Although these techniques successfully identify attacks, they have limits since they are focused on word-level strategies. For instance, word-level methodologies are not able to extract any valuable information from newly discovered words that appear in the test phase and are absent from the training set. In addition, memory problems also arise when the number of distinct words increases.

Rong et al. applied a character-level embedding technique in their proposed CNN model [44]. They used only the query part of HTTP requests to detect injected attacks and conduct the experiments on the data that have been independently crawled by the authors themselves. In addition to the query parameters, Odumuyiwa and Chibueze used the body parameter of POST requests in their character-level CNN model to identify HTTP injection attacks [45]. Character-level CNN was used by Saxe and Berlin to identify malicious URLs, file paths, and registry keys [3]. Saxe and Berlin used several convolutional layers with various kernel sizes followed by a sum-pooling layer in the model. The evaluation is conducted by utilizing Saxe and Berlin custom data.

Jemal et al. used both CNN and LSTM to identify web attacks [2, 19]. An LSTM is included after the CNN layer in the suggested models. The CNN component ignores the irrelevant data and achieves the input's important properties, while the LSTM component captures the data's sequential relationship. While Gong et al. applied character embedding to URLs of the CSIC 2010 dataset, Jemal et al. employed ASCII embedding (the code-level information) of the full text (whole content) of HTTP requests.

Vinayakumar et al. investigated some deep learning algorithms based on CNN, RNN, LSM, and CNN-LSTM architectures to categorize malicious/benign URLs using the character-level embedding technique [46]. LSTM and CNN-LSTM models are the most effective ones for the attack detection task according to the Vinayakumar et al. reported results.

Hung et al. leveraged both character embedding and word embedding techniques to enhance the performance of the proposed CNN model [47]. The performance of the suggested approach is assessed using the URLs of Hung et al.'s self-collected data.

Kasim uses SVM to detect DDoS attacks and makes use of an autoencoder for feature learning along with dimensionality reduction [48]. Yi et al. evaluated and analyzed the application of deep learning-based approaches for network attack detection [49]. Their research covers technologies for feature extraction, traffic representation, model training, and model robustness enhancement, as well as several difficulties and issues that may arise during the development stage, such as unbalanced data and distribution shift.

To represent features and detect anomaly-based web attacks, Pillai and Sharma used deep learning methodologies [50]. A stacked autoencoder (SAE) and a denoising autoencoder (DAE) outputs are combined and fed into the generative adversarial network (GAN) as input to enhance the feature representation. For the classification phase, the deep Boltzmann machine with Bi-LSTM is proposed. As a binary classifier, the deep Boltzmann machine is utilized to detect attacks. Bi-LSTM is additionally applied as a multi-class classifier to categorize various types of attacks.

Thajeel et al. carried out a thorough literature review of ML and deep learning techniques used for the goal of detecting XSS attacks [51]. CNN is found to be the deep learning-based algorithm that is most frequently used. Different preprocessing methods including feature engineering and data cleansing are also examined. In addition, the widely employed performance metrics are assessed. According to Thajeel et al.'s study, accuracy, precision, recall, and F_1 scores are the most commonly employed metrics for the XSS attack detection task.

Dilated CNNs were recently used by Rizvi et al. for an intrusion detection system [52]. Numerous successive dilated CNN layers without including any max-pooling layer are applied along with some feature engineering in the proposed model. The performance is assessed using the CSE-CIC-IDS2018 and CIC-IDS2017 datasets.

Table 1 provides a summary of the studies that have been discussed in this section.

3. Methodology

This section gives details about the dataset in the subsection Dataset and the structure of the proposed methodology as MC-MLDCNN in the subsection Character-Level MC-MLDCNN.

3.1. Dataset.

3.1. Dataset. The CSIC 2010 HTTP dataset is one of the most well-known and frequently used datasets in the area of web security. This dataset is the focus of numerous comparative experiments in the literature [2, 36, 39, 53, 54]. It is created by the Spanish Research National Council (CSIC) in 2010 at the Information Security Institute. It contains the most serious attacks that target the web servers as static attacks and dynamic attacks such as SQL injection, CRLF injection, cross-site scripting (XSS), buffer overflows, information gathering, file disclosure, server-side include, parameter tampering, and unintentional illegal requests.

It has 61065 requests which include 36,000 normal requests and approximately 25,000 abnormal requests. Nearly, 59% of the dataset is normal and 41% is abnormal. The full text of the HTTP requests is used for the experiments in this study. An example of a request is demonstrated in Figure 2.

3.2. Character-Level MC-MLDCNN. The character-level embedding technique and the MC-MLDCNN structure are described in the following subsections.

3.2.1. Character-Level Embedding. A character-level embedding is used to represent and embed the full text of HTTP requests into the model. Character embedding not only makes the model learn the structural patterns of the input string but also gives the model the ability to attain embedding for new unseen inputs. Many word-level embedding approaches suffer from the inability to extract patterns

Table 1: Summary of the related work.

Study	Architecture	Dataset	Remarks	Attack type	Performance
Mehta et al. [28]	Logistic regression, random forest, SVM, and more	Self-collected	Evaluation of supervised/unsupervised ML algorithms (logistic regression achieves the best performance)	SQL injection	Acc: 93.21, Rcll: 77.38, Prec: 100
Louk and Tama [29]	Bagging ensemble of gradient boosting decision trees	HIKARI-2021, **NSL-KDD**, UNSW-NB15	Bagging ensemble of GBM performs the best	Intrusion detection	Acc: 91.57, Rcll: 86.18, Prec: 98.67, F_1: 91.50
Althubiti et al. [30]	LSTM	CSIC 2010	Manual feature extraction is applied	Intrusion detection	Acc: 99.97, Rcll: 99.50, Prec: 99.50
Yin et al. [32]	RNN	NSL-KDD	Feature extraction is performed automatically	Intrusion detection	Acc: 81.29
Xing et al. [33]	LSTM + bidirectional RNN	Real-world datasets	Feature extraction is performed automatically	Cyberattack rate	MSE: 3,628,266, MAD: 463.2715, PMAD: 0.012, MAPE: 0.013
Kasim [34]	Sparse autoencoder + principal component + light gradient boosted machine	ISCX-URL	Sparse autoencoder + principal component are applied for feature learning; light gradient boosted machine is used for feature selection and classification	Phishing attacks	Acc: 99.6, F_1: 99.58, FPR: 0.001
Dawadi et al. [35]	Layered LSTM	Self-collected	Manual feature extraction is performed by analyzing attack-indicator features of IDS ISCX 2012, 2019 DDoS CIC, and CISC 2010 datasets	DDoS attack, XSS and SQL injection	Acc_{DDoS}: 97.57, $\text{Acc}_{\text{XSS/SQL}}$: 89.34
Hao et al. [36]	Bi-LSTM	CSIC 2010	Word2vec is applied for feature representation	Web attack	Acc: 98.35, Rcll: 98.17, Prec: 99.00, F_1: 98.58, FPR: 0.014
Alaoui and Nfaoui [38]	Ensemble of LSTMs	CSIC 2010	Word2vec is applied for feature representation	Web attack	Acc: 78.95, Rcll: 78.41, Prec: 81.54, F_1: 77.57
Zhang et al. [39]	CNN	CSIC 2010	Word-level embedding is applied	Web attack	Acc: 93.35, Rcll: 96.49, FPR: 0.0137
Tian et al. [40]	M-ResNet + FastText	**CSIC 2010**, FWAF, HttpParams dataset	Concatenation of Word2vec and TF-IDF is used for feature vectors. M-ResNet is applied for feature discrimination purpose. Classification is performed using a FastText classifier	Web attack	Acc: 99.41, Rcll: 98.91, DRN: 99.55, F_1: 77.57
Luo et al. [43]	Ensemble of M-ResNets, LSTM, and CNN	**CSIC 2010** real-world dataset	Concatenation of Word2vec and TF-IDF is used for feature representation	Web attack	Acc: 99.47, Rcll: 99.29, Prec: 99.70, FPR: 0.0033
Rong et al. [44]	CNN	Self-collected	Character-level embedding is applied	Injection attacks	Prec: 100, Rcll: 99.7, FPR: 0.0002
Odumuyiwa and Chibueze [45]	CNN	ECML/PKDD 2007, CSIC 2010	Character-level embedding is applied	HTTP injection attacks	Acc: 96.39, Prec: 98.83, Rcll: 95.00, F_1: 97.00, FPR: 0.020
Saxe and Berlin [3]	CNN	Self-collected	Character-level embedding is applied	Malicious URLs, paths, registry keys	AUC_{URL}: 99.30, $\text{AUC}_{\text{FilePath}}$: 97.80, $\text{AUC}_{\text{RegistryKeys}}$: 99.20
Gong et al. [19]	CNN + LSTM	CSIC 2010	Character-level embedding is applied	Web attack	Acc: 97.79, Prec: 98.54, Rcll: 96.04, F_1: 97.27
Jemal et al. [2]	CNN + LSTM	CSIC 2010	ASCII-level embedding is applied	Web attack	Acc: 99.25, Prec: 97.73, Rcll: 99.35, F_1: 98.53

Table 1: Continued.

Study	Architecture	Dataset	Remarks	Attack type	Performance
Vinayakumar et al. [46]	CNN, RNN, LSM, CNN-LSTM, and more	Self-collected	Character-level embedding is applied. The most effective models are LSTM and CNN-LSTM	Malicious URL	Acc_{LSTM}: 99.95, AUC_{LSTM}: 99.99, $\text{Acc}_{\text{CNN-LSTM}}$: 99.96, $\text{AUC}_{\text{CNN-LSTM}}$: 99.99
Hung et al. [47]	CNN	Self-collected	Character-level and word-level embeddings are used	Malicious URL	AUC: 99.29
Kasim [48]	Autoencoder + SVM	**CICIDS**, NSL-KDD, virtual traffic DDoS attack	Autoencoder is used for feature learning and dimensionality reduction. SVM is used for classification	DDoS attack	Acc: 99.41, Prec: 99.66, Rcll: 99.67, F_1: 99.67
Yi et al. [49]	Autoencoders, restricted Boltzmann machine, deep belief networks, CNN, and more	The datasets used in the literature are covered	Review of application of deep learning approaches. It covers feature representation, model training, model robustness enhancement techniques, and problems and challenges of developments	Network attacks	—
Pillai and Sharma [50]	Stacked autoencoder (SAE) + denoising autoencoder (DAE) + generative adversarial network (GAN) + deep Boltzmann machine + Bi-LSTM	CSIC 2010v2	Concatenated form of SAE and DAE outputs is fed into a GAN for feature representation. The deep Boltzmann machine is used to identify attacks. For identifying the different types of attacks, Bi-LSTM is used	Web attack	Prec: 98.78, Rcll: 98.78, F_1: 98.78
Thajeel et al. [51]	ML, deep learning	Frequently used datasets are covered	Literature review of ML and deep learning methodologies and advancements	XSS attacks	—
Rizvi et al. [52]	Dilated CNN	**CSE-CIC-IDS2018**, CIC-IDS2017	Manual feature selection is applied	Intrusion detection	Acc: 99.98

DRN: the percentage of all normal requests that are classified as normal; MSE: mean square error; MAD: mean absolute deviation; PMAD: percent mean absolute deviation; MAPE: mean absolute percentage error. Acc, Prec, Rcll, F_1, and FPR: accuracy, precision, recall, F_1 score, and false-positive rate, respectively. The overall performance of relative datasets is given. In studies containing multiple dataset evaluations, the provided performance is related to the bold dataset.

```
GET http : //localhost : 8080/tienda1/index.jsp HTTP/1.1
User–Agent : Mozilla/5.0 (compatible; Konqueror/3.5; Linux)
KHTML/3.5.8 (like Gecko)
Pragma : no–cache
Cache–control : no–cache
Accept : text/xml, application/xml, application/xhtml+xml, text/html;
q=0.9, text/plain; q=0.8, image/png, */*; q=0.5
Accept–Encoding : x-gzip, x-deflate, gzip, deflate
Accept–Charset : utf–8, utf–8; q=0.5, *; q=0.5
Accept–Language : en
Host : localhost : 8080
Cookie : JSESSIONID=1F767F17239C9B670A39E9B10C3825F4
Connection : close
```

Figure 2: The full text of an HTTP request in the CSIC 2010 dataset.

for unseen words. Besides, the model size will increase as the data size increases. Character-level embedding has the additional benefit of maintaining the model size stable as the number of characters is constant. Consequently, the memory problem regarding word embedding is alleviated. In the embedding phase, a vocabulary consisting of alphabets and numeric characters is formed. In addition, some other characters that appear frequently in HTTP requests are added to the vocabulary as illustrated in Table 2.

The UNK token is added to the vocabulary for the characters that are not in the alphanumeric and special characters defined in Table 2. Token PAD is defined for padding purpose. Every character has a special embedding vector, and this information is kept in an embedding matrix (EM). After getting the indices of the vocabulary according to Table 2, each of the HTTP requests can be represented as a sequence of indices. These indices are the index of each character in the embedding matrix (EM). The sequence length is set to a threshold θ. Any HTTP requests less than this threshold are padded, and the longer ones are truncated. Each character is embedded into a k-dimension vector. The embedding is subsequently given a random initialization before being trained. Each row includes a character's vector representation in the character-level embedding matrix. Hence, a character-level embedding of an input results in a matrix with θ rows and k columns Input $\longrightarrow R^{\theta\times k}$ (see Figure 3).

Table 2: Vocabulary table of characters and their indices in the one-hot encoding matrix.

Column	Index
abcdefghijklmnopqrstuvwxyz	1–26
0123456789	27–36
-,;.!?:' "/\ | _@#$%ˆ&*~'+-=<>()[]{}	37–69
UNK	70
PAD	0

It contains frequently used alphanumeric and special characters in the requests (indices 1–69). Any other character is defined as an unknown character (UNK with index 70). The 0 index is used for padding purpose.

3.2.2. *MC-MLDCNN Framework.*

CNN is a deep network structure made up of several layers such as the input layer, convolutional layer, pooling layer, fully connected layer, and output layer [55]. The alternating convolutional and pooling layers make up the most noticeable structure among these layers. In a multilayer CNN structure, a convolutional layer plus a pooling layer may extract important features at various levels. In CNNs, three architectural principles are integrated to ensure shift invariance at some level: local receptive fields, spatial/temporal subsampling, and shared weights [55]. The relationship between the receptive field size and the number of layers and kernel size is linear. To cover a longer sequence, a bigger receptive field is needed. A bigger receptive field necessitates more layers, which results in complicating the learning process. Dilated convolutions provide receptive fields that are expanding exponentially while maintaining resolution and coverage.

Dilated convolutions [23] are convolutions in which the filter is applied over a region that is longer than its length by ignoring input data at a certain phase. To put it simply, dilated convolutions are convolutions applied to input with specified gaps. The goal of dilated convolutions is to increase the convolution kernel's receptive field while maintaining the number of kernel parameters untouched. It is performed by filling a fixed element 0 between the original convolution kernels. Conventional CNNs are equal to dilated CNNs with a dilation size of 1 without any gap between the parameters of the kernel. An example of a transformed kernel in dilated CNNs is shown in Figure 4.

The receptive field of dilated convolutions can be exponentially enlarged by applying multiple convolutional layers with increasingly dilated values successively. As a result, information in the larger context can be integrated with less computing effort. Figure 5 demonstrates a three-layer convolution structure for both traditional and dilated convolution neural networks. The number of parameters in both CNN and dilated CNN is the same. Under identical circumstances, the dilated CNN may gather data from a wider area of input in comparison to the traditional CNN.

In this study, a multichannel multilayer dilated CNN is proposed as MC-MLDCNN. Each channel is made up of the input layer, dilated convolutional layers each followed by a pooling layer, a fully connected layer, and an output layer. All the convolutional operations are based on 1D convolutions [56]. A multichannel multilayer dilated CNN includes multiple channels c_i (where $i \in \{1, 2, 4, \ldots\}$). In each

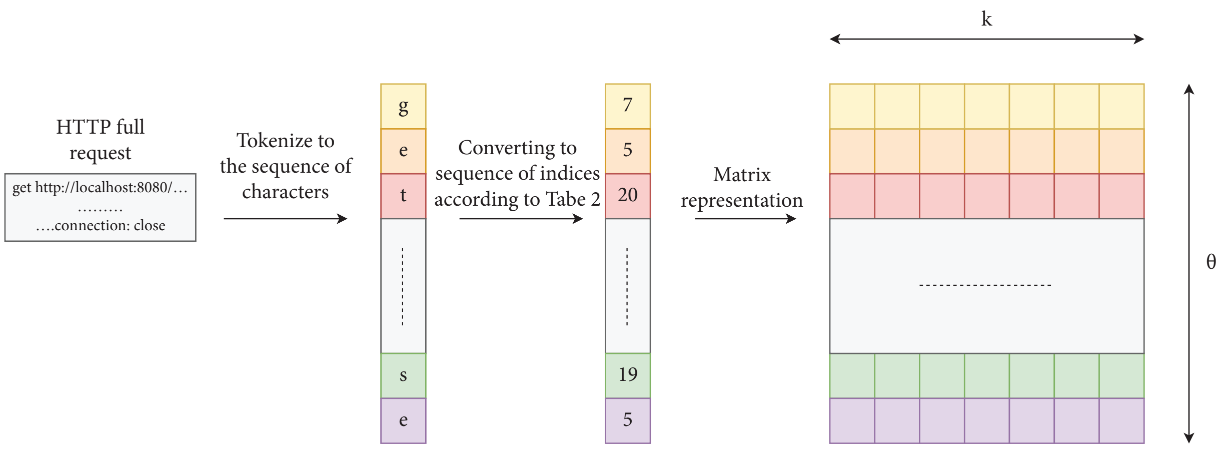
HTTP full request
get http://localhost:8080/...
....
....connection: close
Tokenize to the sequence of characters
g
e
t
s
e
Converting to sequence of indices according to Tabe 2
7
5
20
19
5
Matrix representation
k
θ

Figure 3: Character-level embedding of an HTTP request.

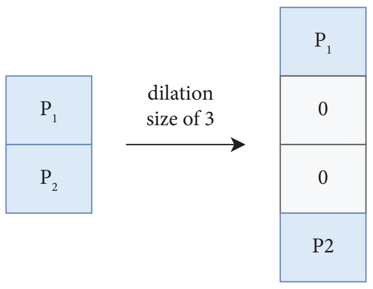

Figure 4: A transformed convolution kernel with a dilation size of 3.

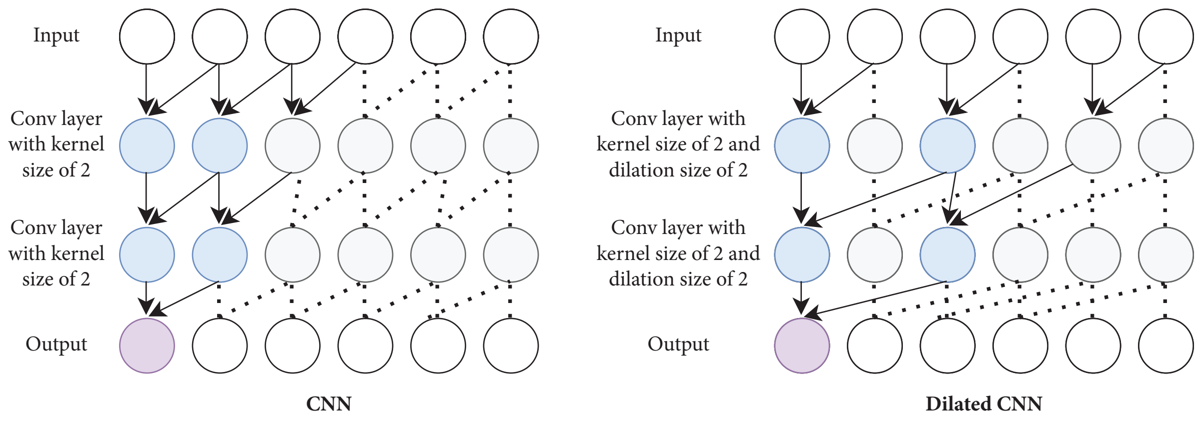
Input
Conv layer with kernel size of 2
Conv layer with kernel size of 2
Output
CNN
Input
Conv layer with kernel size of 2 and dilation size of 2
Conv layer with kernel size of 2 and dilation size of 2
Output
Dilated CNN

Figure 5: Dilated CNNs in comparison with traditional CNNs. A wider range of the input is included in dilated CNN in contrast to the CNN.

channel, c_i contains a fixed kernel size k_i and exponentially increasing dilation size $\in \{1, 2, 4, \ldots\}$. The kernel size varies among the channels $k_i : \in \{2, 3, 4, \ldots\}$. Each of the dilated CNN layers is followed by a max-pooling layer to extract the influential features. The kernel size of max-pooling layers k_p is fixed and set to 3 in all layers.

Each channel can exploit the temporal relationship of length k_i in its input and expand it exponentially. This strategy makes the model able to capture temporal and local dependencies of various scales at different levels. The output of the channels is then concatenated, flattened, and passed to a fully connected layer. Consequently, the model is better able to extract aggregated contextual information at different levels and can learn complex and long-term dependencies.

Given an HTTP request as input, Figure 6 briefly describes the feature mapping phase of a multilayer dilated CNN. Figure 7 illustrates the general structure of MC-MLDCNN.

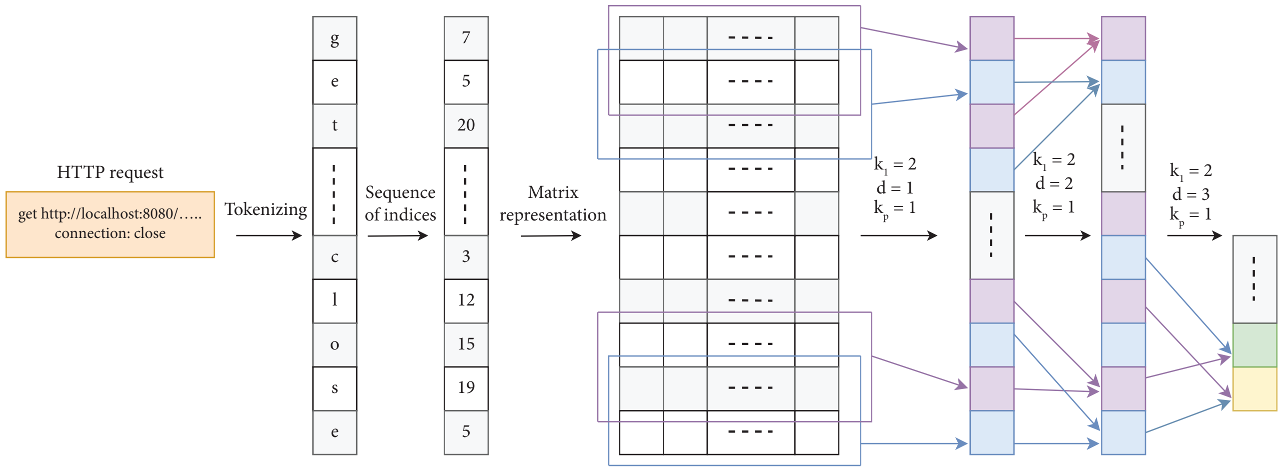

FIGURE 6: Given an HTTP request as input, the feature mapping operation in a multilayer dilated CNN is simply demonstrated. The filter size k_1 is fixed to 2 in all layers. The 1 value of k_1 represents the channel's number which is 1 in this example. The dilation sizes for the first, second, and last layers are 1, 2, and 3, respectively.

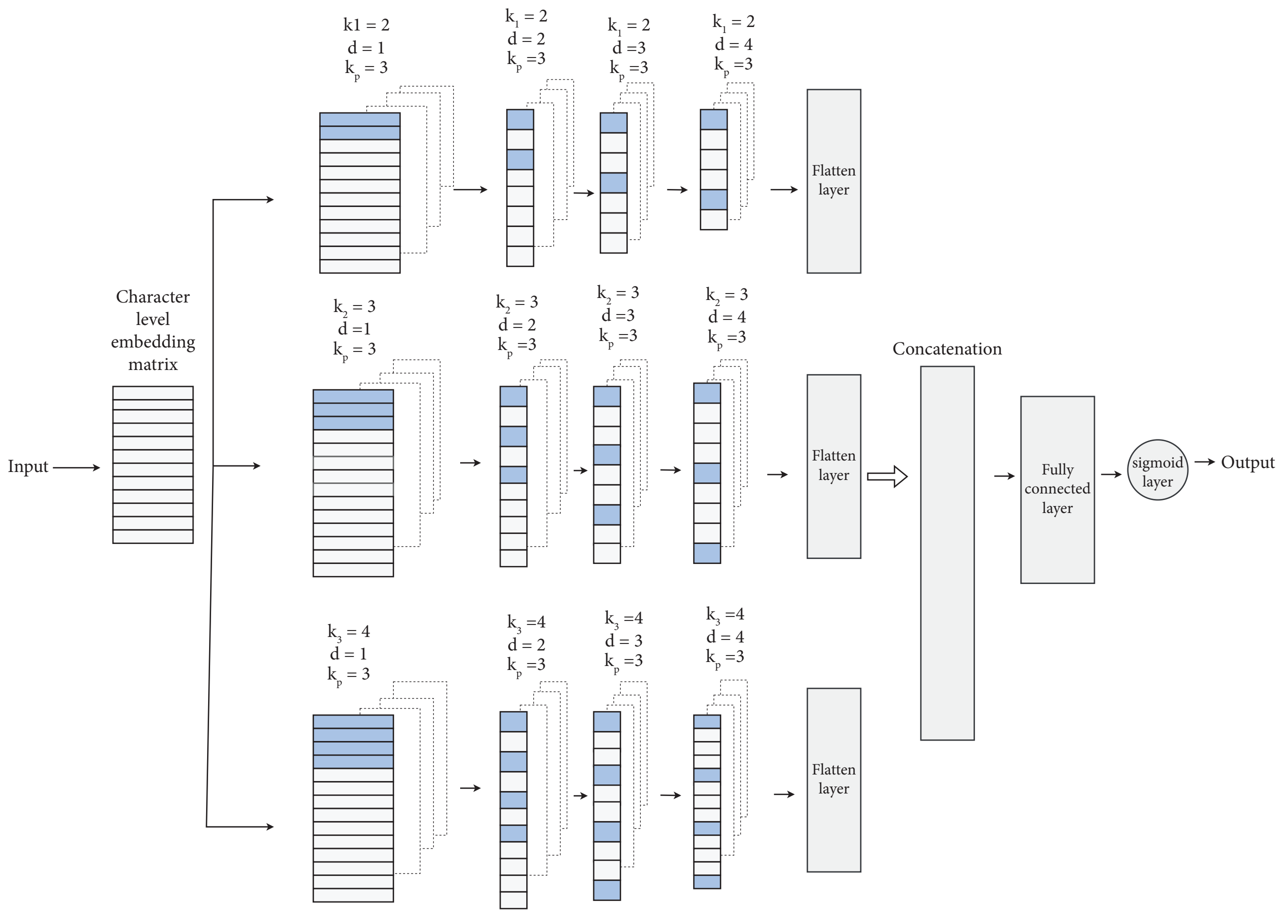

FIGURE 7: The general structure of MC-MLDCNN: the model has three channels. The first channel has a kernel size of 2 with exponentially increasing dilation sizes as 1, 2, and 4. The second channel has a kernel size of 3 with the same dilation sizes. The third layer has a kernel size of 4 with the same dilation sizes as the other layers. Since the kernel sizes vary at each channel, the affected regions are different at each layer. For the sake of simplicity, the dilation sizes are increased incrementally rather than exponentially.

3.2.3. Model Configuration. The model contains two channels. All of the models are implemented with Keras [57] and Python [58]. After converting the HTTP requests to lower cases, a tokenizer is initialized and fitted to the data in the character embedding phase according to Table 2. Each character is embedded into a 71-dimensional vector. The embedding is then randomly initialized and learned during the training phase. The obtained representations are stored in an embedding matrix (EM) where each row is a vector representation of a character. Since more than 99 percent of requests have lengths less than 900, θ is set as 900. Hence, a character-level embedding of each HTTP request results in a matrix with 900 rows and 71 columns. The model consists of 2 channels each with 256 kernels and fixed sizes of 5 and 6, respectively. Each channel includes 3 convolutional layers. The first convolutional layer has no dilation (equal to a dilation size of 1), whereas the next convolutional layers are with successive dilation sizes of 2 and 4. Each convolutional layer is followed by a max-pooling layer with a kernel size of 3. The outputs of each channel are concatenated, flattened, and passed through a fully connected layer with 512 nodes and regularized by the dropout technique at a rate of 0.2. Binary cross entropy as a loss function and the Adam optimizer with a learning rate of 0.001 as an optimization strategy are applied. The ReLU activation function is used in all layers, and the sigmoid layer is applied for classification purpose. A simplified diagram of the model is shown in Figure 8.

4. Results and Discussion

The dataset is divided into training sets and test sets randomly, making sure that the class distribution in each set is similar to the original dataset. The hold-out strategy [30, 59] is applied as the size of the dataset is sufficiently large. Hence, in the experiments, 67% of the dataset is utilized as a training set and 33% is employed as a test set. As a result, a significant portion of the data is allocated to testing while still leaving an adequate amount for the training phase. The experiments are conducted in two steps: the hyperparameter tuning part and the evaluation phase with the obtained parameters.

4.1. Hyperparameter Tuning. The key hyperparameters of the proposed methodology are the number of channels, the number of kernels, the sizes of the kernels, the number of fully connected layers' nodes, and the sizes of the dilations. 10% of the training set is used as the validation set for the assessment's findings. First, a multichannel CNN (MC-CNN) is built as a baseline model to determine the number of channels, with each channel having one block of CNN. The number of kernels and the number of nodes are initialized as 256 and 512, respectively. The findings for various channel sizes and their corresponding kernel sizes are displayed in Table 3. Two channels with kernel sizes of 5 and 6 produce the greatest results. This means the best results are achieved when 256 filters are applied to 5 characters at a time in the first channel and 6 characters in the second channel, respectively. As a result, the number of channels is set to two channels with kernel sizes of 5 and 6.

Further experiments are conducted to obtain the best number of filters (kernels). The outcomes are displayed in Table 4. 256 kernels produce the best accuracy result. The number of kernel parameter (f) is therefore set to 256.

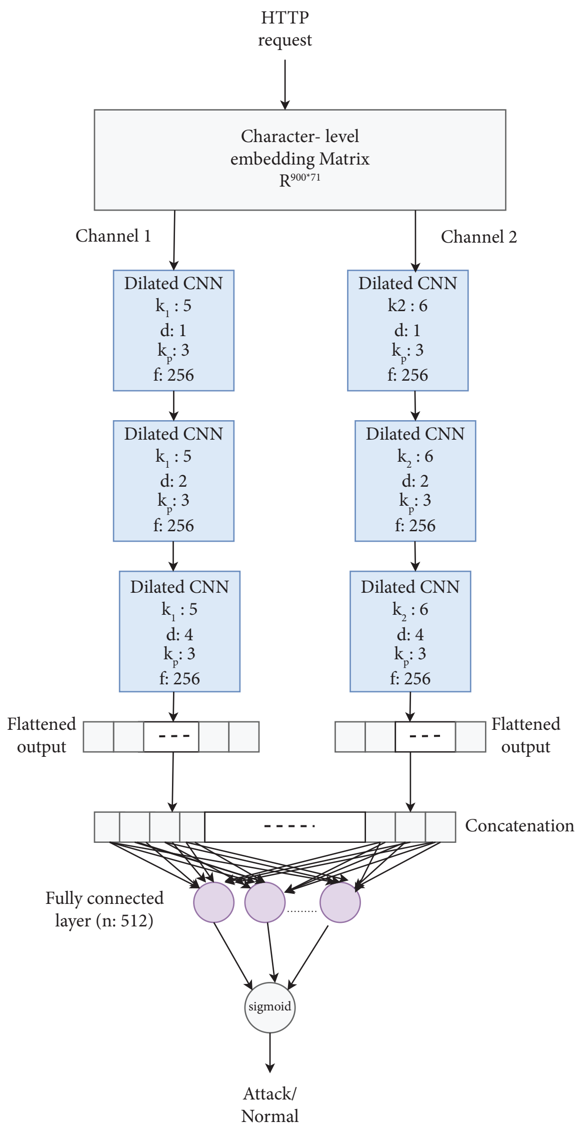
HTTP request
Character- level embedding Matrix R^900*71
Channel 1
Channel 2
Dilated CNN k1 : 5 d: 1 kp: 3 f: 256
Dilated CNN k2 : 6 d: 1 kp: 3 f: 256
Dilated CNN k1 : 5 d: 2 kp: 3 f: 256
Dilated CNN k2 : 6 d: 2 kp: 3 f: 256
Dilated CNN k1 : 5 d: 4 kp: 3 f: 256
Dilated CNN k2 : 6 d: 4 kp: 3 f: 256
Flattened output
Flattened output
Concatenation
Fully connected layer (n: 512)
sigmoid
Attack/ Normal

FIGURE 8: MC-MLDNN model with the relative hyperparameters.

TABLE 3: The number of channels and their relative kernel sizes.

Channels and kernel sizes	Accuracy
[3, 4]	97.97
[4, 5]	98.19
[3–5]	98.22
[5, 6]	**98.24**

[5, 6] show two channels, one with a kernel size of 5 and the other with a kernel size of 6, for instance. The accuracy score with the highest value is bolded.

Table 5 demonstrates the findings for a different number of nodes. The finest results are achieved for 512 nodes. Therefore, the number of nodes parameter is set to 512.

The dilated CNN layers are then added to each channel and optimized. Table 6 represents the results for numerous dilation sizes. The highest accuracy results are achieved for dilation sizes of 1, 2, and 4, respectively.

Table 4: The obtained results for different numbers of kernels.

Number of kernels	Accuracy
64	98.04
128	98.00
256	**98.24**

The accuracy score with the highest performance is bolded.

Table 5: The attained findings for different numbers of nodes.

Number of nodes	Accuracy
64	97.85
128	98.14
256	98.22
512	**98.24**

The accuracy score with the highest performance is bolded.

Table 6: The achieved outcomes for different dilation sizes.

Dilation size	Accuracy
[1, 2]	98.90
[1, 2, 4]	**99.22**
[1, 2, 4, 8]	98.68
[1, 2, 4, 8, 16]	99.10

[1, 2, 4] represent exponentially increasing dilation sizes as 1, 2, and 4, respectively, for each channel. The accuracy score with the highest performance is bolded.

Consequently, the proposed model (MC-MLDCNN) includes two channels with kernel sizes of 5 and 6, respectively. For each channel, there are fixed 256 filters. The fully connected layer's node value is assigned to 512.

4.2. Evaluation Results. Two models are built to examine the effectiveness of dilated CNNs: one as a multichannel CNN (MC-CNN) and the other as a multichannel multilayer CNN (MC-MLCNN). The multichannel CNN model only has one block of CNN in each channel without any dilation. The multichannel multilayer CNN model includes multiple successive CNN blocks in each channel and does not incorporate any dilation as well. The structure is similar to the MC-MLDCNN with a dilation size of 1 for all CNN blocks. In both multichannel CNN and multichannel multilayer CNN, max-pooling is integrated after each CNN layer. All models have the same kernel sizes, number of channels, and other hyperparameters as the proposed MC-MLDCNN model. The performance of the models is shown in Figure 9.

It is clear that layered CNNs, either dilated or not, outperform single-layer CNNs, concluding that utilizing several layers of CNNs in each channel improves the performance of MC-CNN. Dilated CNNs increase the accuracy and precision metrics as a result of exponentially increasing dilation sizes. A higher recall score of nondilated multilayer CNNs implies that attacks are better detected, yet normal requests are misclassified. This means more interruption and less availability for rightful users. The F_1 score provides a more accurate picture of the performance of the models as both recall and precision scores are taken into account. The proposed MC-MLDCNN achieves the best F_1 value. The

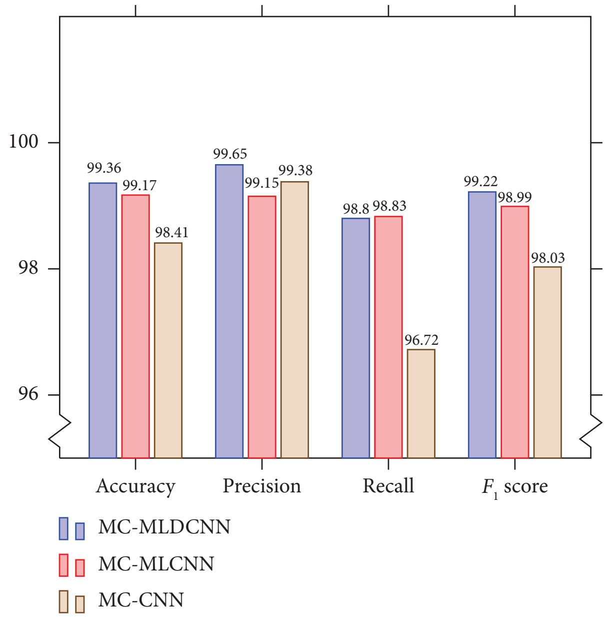

Figure 9: Precision, recall, and F_1 score results of MC-MLDCNN, MC-MLCNN, and MC-CNN.

proposed model outperforms both candidates concerning accuracy and F_1 scores. This indicates that the proposed model suitably distinguishes attacks and normal requests.

The proposed model is compared with the best-performing related work by Rizvi et al. [52]. Rizvi et al.'s model uses a single channel with many layers of dilated CNNs. The size of the dilations has been increased exponentially akin to the proposed model. On the other hand, Rizvi et al. did not include any max-pooling layer. Table 7 shows the comparison of MC-MLDCNN and Rizvi et al.'s model. Another remarkable observation is the fact that MC-MLDCNN rapidly converges in contrast to Rizvi et al.'s model. The accuracy and loss curves for the training and validation sets are shown in Figure 10.

As it can be concluded from Table 7 and Figure 10, Rizvi et al.'s state-of-the-art method performs well. Still, MC-MLDCNN enhances the performance by 2.51%, 2.03%, 4.16%, and 3.11% in terms of accuracy, precision, recall, and F_1 scores, respectively. This enhancement occurs in 4.29% less number of epochs.

Table 8 compares and contrasts the proposed model's performance with the state-of-art works such as a word-level Bi-LSTM-based model proposed by Hao et al. [36], an ASCII-level CNN-LSTM-based method by Jemal et al. [2], a character-level CNN-LSTM-based approach by Gong et al. [19], and a character-level multichannel CNN-based model suggested by Odumuyiwa and Chibueze [45].

Table 8 states that MC-MLDCNN outperforms Hao et al., Gong et al., and Odumuyiwa and Chibueze models in terms of all the assessment metrics. The recall score of the Jemal et al. model is the highest, while its precision score is the lowest. Similar to the previous discussion, this implies misclassification of valid requests. Considering the F_1 scores, MC-MLDCNN demonstrates priority over Jemal et al.'s model.

Table 7: Comparative results for the proposed model and dilated CNN proposed by Rizvi et al.

Methods	Accuracy	Precision	Recall	F_1 score
MC-MLDCNN	**99.36**	**99.65**	**98.80**	**99.22**
Rizvi et al.	96.85	97.62	94.64	96.11

The highest score relative to its assessment metric is bolded.

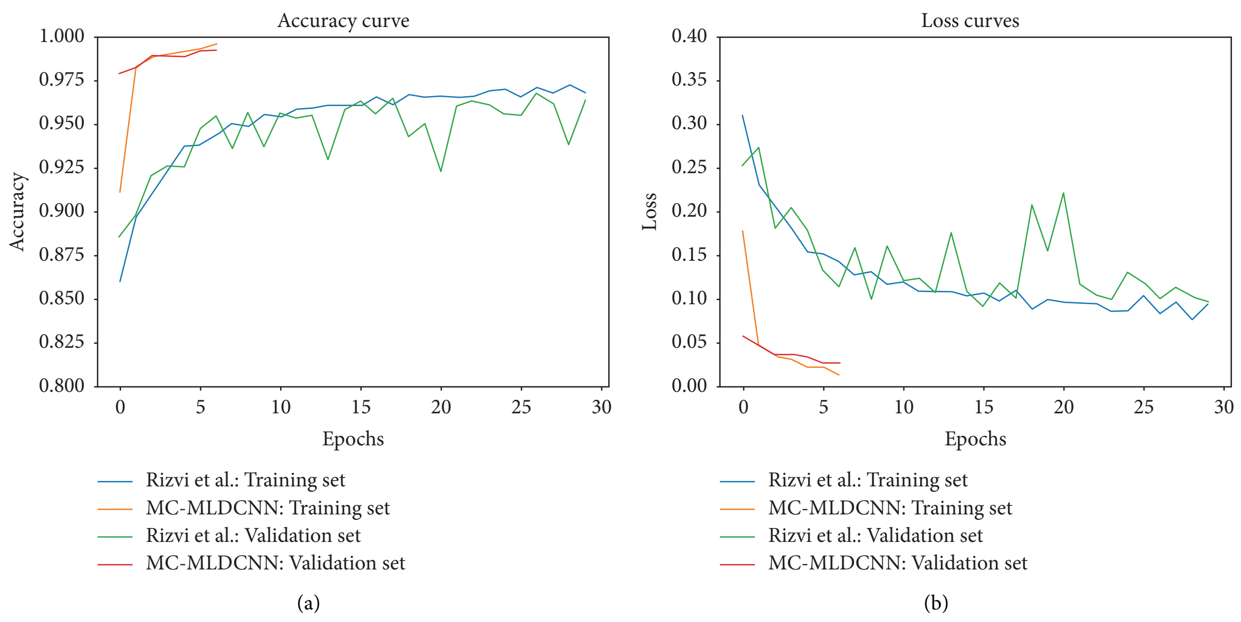

Figure 10: The loss and accuracy curves of training and validation sets of MC-MLDCNN and Rizvi et al. model. (a) The accuracy curves. (b) The loss curves.

Table 8: The reported experimental results for the CSIC 2010 dataset.

Methods	Accuracy	Precision	Recall	F_1 score
Proposed MC-MLDCNN	**99.36**	**99.65**	98.80	**99.22**
Hao et al.	98.35	99.00	98.17	98.58
Jemal et al.	99.25	97.73	**99.35**	98.53
Gong et al.	97.79	98.54	96.04	97.27
Odumuyiwa and Chibueze	96.39	98.83	95.0	96.88

The highest score relative to its evaluation metric is bolded.

In addition to the proposed model with the optimal hyperparameters, 6 other MC-MLDCNN models with various parameter values are trained and evaluated to demonstrate the effectiveness of the proposed methodology in Table 9.

(i) Model 1 is trained using 2 channels with kernel sizes of $k_1 = 5$ and $k_2 = 6$. Each channel consists of 5 layers with dilation sizes of 1, 2, 4, 8, and 16, respectively ($d = [1, 2, 4, 8, 16]$). The number of nodes of the fully connected layer is set to 512 ($n = 512$).

(ii) Model 2 is trained using 3 channels with kernel sizes of $k_1 = 3$, $k_2 = 4$, and $k_3 = 5$. Each channel consists of 3 layers with dilation sizes of 1, 2, and 4, respectively ($d = [1, 2, 4]$). The number of nodes of the fully connected layer is set to 256 ($n = 256$).

(iii) Model 3 is trained using 2 channels with kernel sizes of $k_1 = 5$ and $k_2 = 6$. Each channel consists of 3 layers with dilation sizes of 1, 2, and 4, respectively ($d = [1, 2, 4]$). The number of nodes of the fully connected layer is set to 256 ($n = 256$).

(iv) Model 4 is trained using 2 channels with kernel sizes of $k_1 = 5$ and $k_2 = 6$. Each channel consists of 4 layers with dilation sizes of 1, 2, 4, and 8, respectively ($d = [1, 2, 4, 8]$). The number of nodes of the fully connected layer is set to 256 ($n = 256$).

(v) Model 5 is trained using 3 channels with kernel sizes of $k_1 = 4$, $k_2 = 5$, and $k_3 = 6$. Each channel consists of 3 layers with dilation sizes of 1, 2, and 4, respectively ($d = [1, 2, 4]$). The number of nodes of the fully connected layer is set to 256 ($n = 256$).

Table 9: The experimental results of MC-MLDCNN with various parameters.

Approach	Study or MC-MLDCNN with different parameters	Accuracy	Precision	Recall	F_1 score	FPR
MC-MLDCNN with different parameters	Model 1 ($k_1 = 5$, $k_2 = 6$, $d = [1, 2, 4, 8, 16]$, $n = 512$)	99.12	99.39	98.46	98.93	0.42
	Model 2 ($k_1 = 3$, $k_2 = 4$, $k_1 = 5$, $d = [1, 2, 4]$, $n = 256$)	99.29	99.39	**98.88**	99.13	0.42
	Model 3 ($k_1 = 5$, $k_2 = 6$, $d = [1, 2, 4]$, $n = 256$)	**99.34**	**99.76**	98.63	**99.19**	**0.16**
	Model 4 ($k_1 = 5$, $k_2 = 6$, $d = [1, 2, 4, 8]$, $n = 256$)	99.00	99.00	98.56	98.78	0.69
	Model 5 ($k_1 = 4$, $k_2 = 5$, $k_3 = 6$, $d = [1, 2, 4]$, $n = 256$)	**99.34**	99.62	98.75	**99.19**	0.26
	Model 6 ($k_1 = 3$, $k_2 = 4$, $k_3 = 5$, $d = [1, 2, 4]$, $n = 512$)	99.08	99.46	98.30	98.88	0.37
	Proposed ($k_1 = 5$, $k_2 = 6$, $d = [1, 2, 4]$, $n = 512$)	**99.36**	**99.65**	98.80	**99.22**	**0.24**
Competitive deep learning models in the literature	Hao et al. [36]	98.35	99.00	98.17	98.58	1.40
	Jemal et al. [2]	99.25	97.73	**99.35**	98.53	—
	Gong et al. [19]	97.79	98.54	96.04	97.27	—
	Odumuyiwa and Chibueze [45]	96.39	98.83	95.0	96.88	2.00
	Rizvi et al. [52]	96.85	97.62	94.64	96.11	1.60
Classic deep learning models	CNN	97.93	97.96	96.97	97.46	1.40
	LSTM	97.70	97.91	96.46	97.18	1.43
	Bi-LSTM	97.54	97.98	96.00	96.98	1.38

The efficiency of the proposed methodology in comparison to benchmark deep learning models. The top two scores are in bold.

(vi) Model 6 is trained using 3 channels with kernel sizes of $k_1 = 3$, $k_2 = 4$, and $k_3 = 5$. Each channel consists of 3 layers with dilation sizes of 1, 2, and 4, respectively $(d = [1, 2, 4])$. The number of nodes of the fully connected layer is set to 512 $(n = 512)$.

In Table 9, the MC-MLDCNN models are contrasted not only with the competitive deep learning models in the literature but also with the character-level traditional deep learning models like CNN, LSTM, and Bi-LSTM.

All of the MC-MLDCNN models outperform the benchmark deep learning models when comparing precision outcomes. Jemal et al., however, performed the best among the recall scores. The second-best recall values are shared by all of the MC-MLDCNN models. All of the MC-MLDCNN models outperform the benchmark deep learning models when F_1 scores are taken into account. It is a desired situation given that the goal of this research is to accurately detect web attacks.

Except for Jemal et al.'s model, all the MC-MLDCNN models outperform the benchmark deep learning models when accuracy results are compared. Although the proposed model, model 2, model 3, and model 5 outperform Jemal et al.'s accuracy and produce the greatest results, the other MC-MLDCNN models' accuracy scores lag behind Jemal et al.'s. It implies that Jemal et al. performed slightly better in normal request detection tasks than model 1, model 4, and model 6. It should be highlighted that the outcomes with the best hyperparameters of benchmark models are compared with the MC-MLDCNN models. As a result, it is expected that occasionally some of the MC-MLDCNN models perform slightly worse. This implies the importance of selecting appropriate hyperparameters. Even models 1, 4, and 6 surpass all benchmark models when it comes to the task of attack detection, demonstrating the effectiveness of this methodology.

For an even more comprehensive assessment, the FPR scores (the rate of classifying normal requests as attacks) are also added to Table 9. The FPR values are unfortunately not provided in the studies by Jamal et al. and Gong et al. Excluding Jemal et al. and Gong et al., it should be noted that MC-MLDCNN models have the lowest FPR values indicating that they are effective at detecting normal requests as well. It is noteworthy that properly identifying attacks is prior to properly detecting normal requests because of the destructive consequences of a situation in which an attack is mistaken for a normal request as opposed to the one in which a normal request is mistaken for an attack. Since the accuracy and F_1 scores of MC-MLDCNN models are the highest, the proposed methodology has precedence over the models by Gong et al. and Jamal et al. even if the FPR values for Gong et al. and Jamal et al. are not supplied.

The character-level MC-MLDCNN model outperforms a number of cutting-edge models in the literature as well as traditional deep learning models, according to the experimental results. The provided methodology successfully detects web attacks. Although the aim of this research is to reliably identify web attacks, it has also been demonstrated that with the right hyperparameter, the model performs excellently in identifying normal requests as well.

5. Conclusion

Web attacks have severe effects such as data breaches, financial losses, reputational harm, and other consequences for both customers and organizations. Web attack detectors are essential for guaranteeing security, in other words, the confidentiality, integrity, and availability of the systems. The goal is to maintain system availability while protecting data confidentiality and system integrity. Therefore, precise web detectors provide the most reliable systems.

Even though various deep learning approaches are suggested in the literature with acceptable detection performance, many of them struggle to effectively capture complicated and lengthy sequence relationships of HTTP requests. The MC-MLDCNN methodology, which is proposed in this study, is capable of efficiently learning complex and lengthy character relationships of the requests. The method is based on the integration of multichannel and multilayer dilated CNNs. The MC-MLDCNN learns the dependencies among the characters of HTTP requests at various levels and across a broad range. Therefore, it successfully captures the long-term dependency of characters in HTTP requests. Consequently, the model is accurate in recognizing the intricate pattern of attacks.

Several MC-MLDCNN models with different parameter settings are trained and evaluated. The experimental results are contrasted with various prior effective deep learning-based approaches that have been proposed in the literature. The outcomes show that the proposed methodology outperforms the benchmark deep learning models and can reliably identify attacks. In addition, the method's accuracy surpasses all benchmark models when the right hyperparameters are used. This indicates the proposed methodology outperforms the benchmark models in terms of overall performance for both detecting attacks and normal requests. Although developing an accurate web attack detection system is the major objective of this research, a system with a high false positive is ineffective. A false positive, in other words, identifying a normal request as an attack, stops business continuity. This means availability is lost for that moment. Accurately recognizing web attacks and limiting false positives are two factors that must be balanced for a web attack detection system to be successful. The results of the experiments show conclusively that the character-level MC-MLDCNN methodology fits the criteria and is effective for usage in web application security systems.

The proposed methodology has the potential to dramatically improve the capability of identifying complex web attacks. The future goal is to accumulate data regarding tricky and sophisticated attacks that the practically deployed security mechanisms miss. By evaluating the proposed methodology on these samples, future research will primarily concentrate on enhancing the performance of the current model and developing and exploring cutting-edge ways to offer more robust and dependable protection against web attacks.

Data Availability

The benchmark dataset is an open-source dataset and available over the internet.

Ethical Approval

No ethical issues are disclosed by the authors. No personally identifiable data have been used in this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] B. M. P. Waseso and N. A. Setiyanto, "Web phishing classification using combined machine learning methods," *Journal of Computing Theories and Applications*, vol. 1, no. 1, pp. 11–18, 2023.

[2] I. Jemal, M. A. Haddar, C. Omar, and A. Mahfoudhi, "M-cnn: a new hybrid deep learning model for web security," in *Proceedings of the 2020 IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA)*, November 2020.

[3] J. Saxe and K. Berlin, "Expose: a character-level convolutional neural network with embeddings for detecting malicious urls, file paths and registry keys," 2017, https://arxiv.org/abs/1702.08568.

[4] A. D. Santoso, F. B. Cahyono, B. Prahasta, I. Sutrisno, and A. Khumaidi, "Development of pcb defect detection system using image processing with yolo cnn method," *International Journal of Artificial Intelligence Research*, vol. 6, 2023.

[5] E. Parsaeimehr, M. Fartash, and J. Akbari Torkestani, "Improving feature extraction using a hybrid of cnn and lstm for entity identification," *Neural Processing Letters*, vol. 1–16, 2023.

[6] J. Xin, X. Lyu, and J. Ma, "Natural backdoor attacks on speech recognition models," in *Proceedings of theMachine Learning for Cyber Security: 4th International Conference, ML4CS 2022*, pp. 597–610, Springer, Guangzhou, China, December 2022.

[7] A. H. El Hakea and M. W. Fakhr, "Recent computer vision applications for pavement distress and condition assessment," *Automation in Construction*, vol. 146, Article ID 104664, 2023.

[8] C. Han, L. Zhang, Y. Tang, W. Huang, F. Min, and J. He, "Human activity recognition using wearable sensors by heterogeneous convolutional neural networks," *Expert Systems with Applications*, vol. 198, Article ID 116764, 2022.

[9] Y. Tang, L. Zhang, F. Min, and J. He, "Multiscale deep feature learning for human activity recognition using wearable sensors," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 2, pp. 2106–2116, 2023.

[10] W. Huang, L. Zhang, H. Wu, F. Min, and A. Song, "Channel-equalization-har: a light-weight convolutional neural network for wearable sensor based human activity recognition," *IEEE Transactions on Mobile Computing*, vol. 22, no. 9, pp. 5064–5077, 2023.

[11] W. Huang, L. Zhang, S. Wang, H. Wu, and A. Song, "Deep ensemble learning for human activity recognition using wearable sensors via filter activation," *Association for Computing Machinery Transactions on Embedded Computing Systems*, vol. 22, no. 1, pp. 1–23, 2022.

[12] X. Wang, L. Zhang, W. Huang et al., "Deep convolutional networks with tunable speed–accuracy tradeoff for human activity recognition using wearables," *IEEE Transactions on Instrumentation and Measurement*, vol. 71, pp. 1–12, 2022.

[13] Y. Ren, Y. Xiao, Y. Zhou, Z. Zhang, and Z. Tian, "Cskg4apt: a cybersecurity knowledge graph for advanced persistent threat organization attribution," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, 2022.

[14] Y. Chai, L. Du, J. Qiu, L. Yin, and Z. Tian, "Dynamic prototype network based on sample adaptation for few-shot malware detection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 5, pp. 4754–4766, 2022.

[15] M. R. Faisal, I. Budiman, and F. Abadi, "Applying features based on word embedding techniques to 1d cnn for natural disaster messages classification," in *Proceedings of the 2022 5th International Conference of Computer and Informatics Engineering (IC2IE)*, pp. 192–197, IEEE, Jakarta, Indonesia, September 2022.

[16] M. D. Habib, A. Mah, M. D. Kamal, and A. Samad, "Emotion recognition from microblog managing emoticon with text and classifying using 1d cnn," 2023, https://arxiv.org/abs/2301.02971.

[17] Z. Alshingiti, R. Alaqel, and J. Al-Muhtadi, "A deep learning-based phishing detection system using cnn, lstm, and lstm-cnn," *Electronics*, vol. 12, no. 1, p. 232, 2023.

[18] M. N. A. A. Nazarri and M. H. M Yusof, "Web-based intrusion detection system on cnn," *International Journal of Advanced Defence, Security and Maritime Studies*, vol. 2, no. 2, 2022.

[19] X. Gong, J. Lu, and Y. Wang, "Cecor-net: a character-level neural network model for web attack detection," in *Proceedings of the 2019 IEEE International Conference on Smart Cloud (SmartCloud)*, pp. 98–103, IEEE, Tokyo, Japan, December 2019.

[20] A. Gupta and A. M. Rush, "Dilated convolutions for modeling long-distance genomic dependencies," 2017, https://arxiv.org/abs/1710.01278.

[21] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of the International Conference on Machine Learning*, pp. 1310–1318, Pmlr, Westminster, London, May 2013.

[22] B. Rekabdar and C. Mousas, "Dilated convolutional neural network for predicting driver's activity," in *Proceedings of the 2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 3245–3250, IEEE, Maui, HI, USA, November 2018.

[23] Y. Fisher and V. Koltun, "Multi-scale context aggregation by dilated convolutions," 2015, https://arxiv.org/abs/1511.07122.

[24] N. R. Wankhade, K. K. Bhoyar, and A. Bagde, "Semantic segmentation of retinal vasculature using light patch-based dilated cnn," in *Proceedings of the International Conference on Cognitive and Intelligent Computing: ICCIC*, pp. 269–276, Springer, Singapore, January 2021.

[25] D. Jin, R. Kang, H. Zhang, W. Hao, and G. Chen, "Improving abstractive summarization via dilated convolution," *Journal of Physics: Conference Series*, vol. 1616, Article ID 012078, 2020.

[26] P. Ryan, R. Valle, and C. Bryan, "Waveglow: a flow-based generative network for speech synthesis," in *Proceedings of the ICASSP 2019-2019 IEEE International Conference on*

Acoustics, Speech and Signal Processing (ICASSP), pp. 3617–3621, IEEE, Brighton, UK, May 2019.

[27] Csic, "Csic 2010 dataset," 2010, https://www.tic.itefi.csic.es/dataset/.

[28] D. Mehta, H. Suhagiya, and H. Gandhi, "Sqliml: a comprehensive analysis for sql injection detection using multiple supervised and unsupervised learning schemes," *Scaling Number Computer Science*, vol. 4, no. 3, p. 281, 2023.

[29] M. H. L. Louk and B. A. Tama, "Dual-ids: a bagging-based gradient boosting decision tree model for network anomaly intrusion detection system," *Expert Systems with Applications*, vol. 213, Article ID 119030, 2023.

[30] S. Althubiti, W. Nick, J. Mason, X. Yuan, and E. Albert, "Applying long short-term memory recurrent neural network for intrusion detection," in *Proceedings of the SoutheastCon 2018*, pp. 1–5, IEEE, St. Petersburg, FL, USA, April 2018.

[31] S. Althubiti, X. Yuan, and E. Albert, "Analyzing http requests for web intrusion detection," *KSU Proceedings on Cybersecurity Education, Research and Practice*, vol. 2, 2017.

[32] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21954–21961, 2017.

[33] F. Xing, M. Xu, S. Xu, and P. Zhao, "A deep learning framework for predicting cyber attacks rates," *European Association for Signal Processing Journal on Information Security*, vol. 2019, pp. 1–11, 2019.

[34] Ö. Kasim, "Automatic detection of phishing pages with event-based request processing, deep-hybrid feature extraction and light gradient boosted machine model," *Telecommunication Systems*, vol. 78, no. 1, pp. 103–115, 2021.

[35] B. R. Dawadi, B. Adhikari, and D. K. Srivastava, "Deep learning technique-enabled web application firewall for the detection of web attacks," *Sensors*, vol. 23, no. 4, p. 2073, 2023.

[36] S. Hao, J. Long, and Y. Y. Bl-ids, "Detecting web attacks using bi-lstm model based on deep learning," in *Proceedings of the Security and Privacy in New Computing Environments: Second EAI International Conference, SPNCE 2019*, pp. 551–563, Springer, Tianjin, China, April 2019.

[37] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in Neural Information Processing Systems*, vol. 26, 2013.

[38] R. L. Alaoui and E. H. Nfaoui, "Web attacks detection using stacked generalization ensemble for lstms and word embedding," *Procedia Computer Science*, vol. 215, pp. 687–696, 2022.

[39] M. Zhang, B. Xu, S. Bai, S. Lu, and Z. Lin, "A deep learning method to detect web attacks using a specially designed cnn," in *Proceedings of the Neural Information Processing: 24th International Conference, ICONIP 2017*, pp. 828–836, Springer, Guangzhou, China, November 2017.

[40] Z. Tian, C. Luo, J. Qiu, X. Du, and M. Guizani, "A distributed deep learning system for web attack detection on edge devices," *IEEE Transactions on Industrial Informatics*, vol. 16, no. 3, pp. 1963–1971, 2019.

[41] J. Armand, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," 2016, https://arxiv.org/abs/1607.01759.

[42] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, Las Vegas, NV, USA, June 2016.

[43] C. Luo, Z. Tan, G. Min, J. Gan, W. Shi, and Z. Tian, "A novel web attack detection system for internet of things via ensemble classification," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 8, pp. 5810–5818, 2021.

[44] W. Rong, B. Zhang, and X. Lv, "Malicious web request detection using character-level cnn," in *Proceedings of the Machine Learning for Cyber Security: Second International Conference, ML4CS 2019*, pp. 6–16, Springer, Xi'an, China, September 2019.

[45] V. Odumuyiwa and A. Chibueze, "Automatic detection of http injection attacks using convolutional neural network and deep neural network," *Journal of Cyber Security and Mobility*, vol. 9, no. 4, pp. 489–514, 2021.

[46] R. Vinayakumar, K. P. Soman, and P. Poornachandran, "Evaluating deep learning approaches to characterize and classify malicious url's," *Journal of Intelligent and Fuzzy Systems*, vol. 34, no. 3, pp. 1333–1343, 2018.

[47] L. Hung, Q. Pham, D. Sahoo, and S. C. H. Hoi, "Urlnet: learning a url representation with deep learning for malicious url detection," 2018, https://arxiv.org/abs/1802.03162.

[48] Ö. Kasim, "An efficient and robust deep learning based network anomaly detection against distributed denial of service attacks," *Computer Networks*, vol. 180, Article ID 107390, 2020.

[49] T. Yi, X. Chen, Y. Zhu, W. Ge, and Z. Han, "Review on the application of deep learning in network attack detection," *Journal of Network and Computer Applications*, vol. 212, Article ID 103580, 2023.

[50] S. Pillai and D. A. Sharma, "Hybrid unsupervised web-attack detection and classification–a deep learning approach," *Computer Standards and Interfaces*, vol. 86, Article ID 103738, 2023.

[51] I. K. Thajeel, K. Samsudin, S. J. Hashim, and F. Hashim, "Machine and deep learning-based xss detection approaches: a systematic literature review," *Journal of King Saud University- Computer and Information Sciences*, vol. 35, no. 7, Article ID 101628, 2023.

[52] S. Rizvi, M. Scanlon, J. McGibney, and J. Sheppard, "Deep learning based network intrusion detection system for resource-constrained environments," in *Proceedings of the International Conference on Digital Forensics and Cyber Crime*, pp. 1–7, Springer, Switzerland, November 2023.

[53] M. Ito and H. Iyatomi, "Web application firewall using character-level convolutional neural network," in *Proceedings of the 2018 IEEE 14th International Colloquium on Signal Processing and its Applications (CSPA)*, pp. 103–106, IEEE, Penang, Malaysia, March 2018.

[54] J. Wang, Z. Zhou, and J. Chen, "Evaluating cnn and lstm for web attack detection," in *Proceedings of the 2018 10th International Conference on Machine Learning and Computing*, pp. 283–287, Macau, China, February 2018.

[55] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," *The handbook of brain theory and neural networks*, vol. 3361, no. 10, p. 1995, 1995.

[56] K. Yoon, "Convolutional neural networks for sentence classification," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 1746–1751, Doha, Qatar, October 2014.

[57] F. Chollet, "Keras," 2015, https://github.com/fchollet/keras.

[58] G. V. Rossum and F. L. Drake Jr, *Python Tutorial*, Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[59] T. Hastie, R. Tibshirani, J. H. Friedman, and J. H. Friedman, *The Elements Of Statistical Learning: Data Mining, Inference, And Prediction*, Springer, Berlin, Germany, 2009.