National College of Ireland

DoS Attack Detection and Mitigation through Deep Learning Techniques

MSc Research Project
MSCCYB1

Saurabh Sanjayprasad Gupta
Student ID: X20224371

School of Computing
National College of Ireland

Supervisor: Prof. Joel Aleburu

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: Saurabh Sanjayprasad Gupta……

Student ID: X20224371……

Programme: MSCCYB1…… **Year:** 2022-2023

Module: MSc Research Project……

Supervisor: Prof. Joel Aleburu……

Submission Due Date: 18TH September 2023……

Project Title: DoS Attack Detection and Mitigation through Deep Learning Techniques

Word Count: 6043…… **Page Count**: 19……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: Saurabh Gupta ……

Date: 18TH September 2023……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

DoS Attack Detection and Mitigation through Deep Learning Techniques

Abstract

In the realm of cybersecurity, the increasing occurrence of Denial-of-Service (DOS) attacks has presented significant hurdles in ensuring that online services remain consistently accessible. This report focuses on the critical task of identifying and mitigating DOS attacks, employing advanced techniques from the field of Deep Learning. Specifically, we employ a specialized neural network called Convolutional Long Short-Term Memory (CLSTM) as the primary tool for categorizing and detecting DOS attacks. For benchmarking purposes, we also utilize the Support Vector Machine (SVM), a widely used machine learning approach. The study involves a detailed analysis of data from network traffic, which we break down into four distinct classes: Benign (normal activity), MSSQL, Syn (SYN flood attacks), and UDP (UDP flood attacks), each representing different attack scenarios. Utilizing the capabilities of deep learning, we train the CLSTM algorithm to recognize and classify these classes with an exceptional level of accuracy. Our initial experimental findings showcase an impressive detection accuracy of 98.91% using the CLSTM model, reaffirming its effectiveness in addressing DOS attacks. In comparison, the SVM model achieves a detection accuracy of 75.42%, highlighting the superior performance of the CLSTM approach.

Keywords: *Denial-of-Service (DOS) Attacks, Deep Learning, Convolutional Long Short-Term Memory (CLSTM), SVM*

1. Introduction

In today's digital age, the seamless functioning of online services has become an integral part of our daily lives. From communication and e-commerce to entertainment and information sharing, the internet underpins an interconnected global ecosystem. However, this unprecedented level of connectivity also introduces vulnerabilities, with cyber threats and attacks becoming increasingly sophisticated and pervasive. Among these threats, Denial-of-Service (DOS) attacks stand out as a particularly disruptive force, targeting the availability of online services and causing significant financial and reputational losses to businesses and organizations. Over past years, DOS attacks become most common, following chart shows that within the

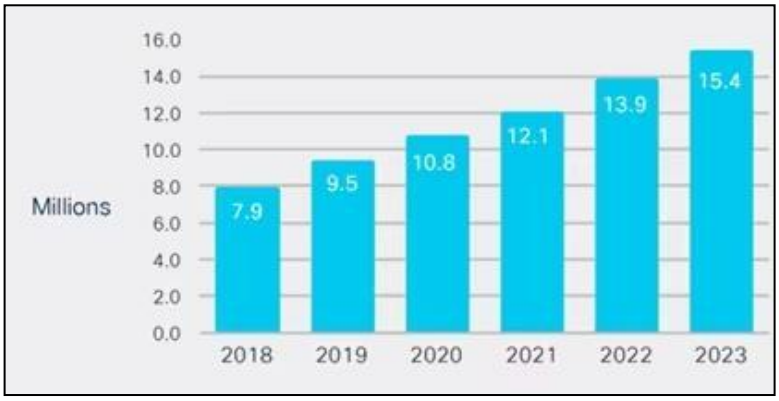


Fig 1. Cisco's analysis on DoS attacks history and predictions[1]

span of 5 years the number of DOS attacks are doubled as analysis done by Cisco's annual internet report.

[1] https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html

Types of DOS Attacks

- **Benign DOS Attack:** DOS, short for "Denial-of-Service," includes "Benign" DOS attacks, arising inadvertently from legitimate user activity or traffic spikes. For example, a website could face temporary unavailability due to heavy traffic during a product launch, without malicious intent.
- **MSSQL DOS Attack:** DOS, short for Denial-of-Service, involves exploiting MSSQL vulnerabilities to inundate Microsoft SQL Server databases with queries, causing server overload, unresponsiveness, and potential data loss.
- **Syn Flood Attack:** DOS means "Denial of Service." A Syn Flood attack is a typical DDoS form targeting TCP's handshake process. Attackers send numerous connection requests (SYN packets) without final acknowledgment (ACK packet), filling the server's queue and causing a denial of service.
- **UDP Flood Attack:** A UDP Flood attack, a form of DDoS attack, focuses on network infrastructure. Using the connectionless UDP protocol, commonly employed in streaming, VoIP, and gaming, attackers inundate the target system with numerous UDP packets. This overwhelms the target's capacity and resources due to the protocol's lack of acknowledgment, causing significant disruption.

The proliferation of DOS attacks has posed substantial challenges to maintaining uninterrupted online service availability. These attacks overload a target system with an overwhelming volume of traffic, rendering it incapable of responding to legitimate user requests. In essence, they disrupt the delicate balance that sustains the digital infrastructure, leading to service downtime and customer frustration. Recognizing the gravity of this issue, the field of cybersecurity has continually evolved to develop innovative methods for detecting and mitigating DOS attacks.

This report is dedicated to a comprehensive exploration of DOS attack detection and mitigation, with a specific focus on leveraging advanced Deep Learning Techniques. Deep Learning, a subset of machine learning, has demonstrated remarkable capabilities in handling complex and unstructured data, making it a promising avenue for addressing the intricate patterns associated with DOS attacks. At the heart of our investigation lies the Convolutional Long Short-Term Memory (CLSTM) neural network, an advanced algorithm designed to capture both spatial and temporal dependencies within data. The CLSTM algorithm is poised to revolutionize the field of DOS attack detection by enabling the automated identification of attack patterns and, subsequently, the implementation of effective mitigation strategies. To gauge the efficacy of the CLSTM algorithm, we employ a benchmarking approach, contrasting its performance with the widely used Support Vector Machine (SVM) classifier.

1.1 Research question and objectives

Research question

- How effective is the Convolutional Long Short-Term Memory (CLSTM) algorithm in accurately detecting and classifying different types of Denial-of-Service (DOS) attacks compared with machine learning algorithm SVM?

Objectives

- Introduce the concept of deep learning and its potential in enhancing cybersecurity, particularly in the context of DOS attack detection.
- Train the CLSTM algorithm on the categorized data to enable accurate classification of DOS attacks.
- Compare the CLSTM algorithm's performance with the Support Vector Machine (SVM) classifier to showcase the advantages of deep learning in DOS attack detection.

2. Related Work

Vishwakarma et al. [1] explored countering distributed denial-of-service (DDoS) attacks in Internet of Things (IoT) and software-defined networking (SDN) systems using machine learning (ML) methods. Nonetheless, this study was confined to IoT and SDN domains, neglecting other potential DDoS attack areas. Furthermore, its scope was limited to the cloud environment, potentially hampering its applicability to diverse network settings. The authors exclusively examined established techniques, disregarding emerging strategies absent from academic literature. While shedding light on ML's role in IoT and SDN DDoS defense, it is imperative to acknowledge this research's limitations, such as its narrow focus and confinement to cloud-based scenarios.

Tripathi et al., [2] conducted a research study focusing on categorizing attacks based on application layer protocols (ALP). While not exclusive to LDoS attacks, their research divided attacks into protocol-specific and generic categories. Protocol-specific encompassed application protocols with specific vulnerabilities, while generic included protocols with general vulnerabilities. Although their study didn't singularly target LDoS attacks, it extensively compared and analyzed attacks and defense mechanisms within these categories. Tripathi and Hubballi's research provides valuable insights into the nature of ALP-related vulnerabilities, aiding the understanding of potential attack vectors and the corresponding defense strategies.

M. S. Essayed et al. [3] examined the utilization of machine learning techniques for identifying and mitigating malicious traffic within Software Defined Networks (SDNs). The authors systematically evaluated prevailing machine learning approaches through benchmarking, subsequently introducing a fortified framework for enhancing attack detection in SDNs. The research emphasizes the heightened susceptibility of SDNs to security vulnerabilities compared to traditional systems, underscoring the indispensability of machine learning methodologies for bolstering their security measures. The study also provides empirical evidence by showcasing experimental outcomes achieved through their proposed framework, involving a collection of publicly available Intrusion Detection Systems (IDSs).

Tang et al. [4], who proposed a density-based application space clustering algorithm known as SADBSCAN. The essence of their approach lies in grouping network traffic and utilizing cosine similarity to ascertain the presence of DoS attacks within a particular cluster. This innovative method capitalizes on the inherent patterns present in network traffic, enabling the algorithm to effectively differentiate between normal and malicious activity. By leveraging the power of clustering and similarity metrics, Tang and his team offer a novel perspective on tackling DoS attacks.

T. Alharbi et al., [5] introduced a new approach to prevent Distributed Denial of Service (DDoS) attacks. Instead of relying solely on traditional methods like dedicated security appliances or cloud-based protection services, they proposed using edge computing and Network Functions Virtualization (NFV). These

alternatives help overcome limitations like hardware constraints, privacy issues, and delays. The authors designed a two-stage framework within NFV and edge computing, making it more efficient compared to old-fashioned ways. This research enriches the literature survey in the field of DDoS mitigation techniques. It provides insight into existing methods, points out shortcomings of traditional and cloud-based approaches, and presents the potential advantages of employing NFV and edge computing for effective DDoS defense.

Osorio et al. [6] delved into the realm of Gaussian Mixture Models (GMM) and Universal Background Models (UBM) to counter DoS/DDoS network attacks. This investigation culminated in the demonstration of the efficacy of GMM and UBM technologies in detecting and mitigating network threats. By combining advanced statistical models with machine learning principles, Osorio and his collaborators contribute to a multi-faceted approach to enhancing network security.

Reddy et al. [7] presented an intriguing integration of the Objection-Based Learning (OBL) technique with the crow search algorithm (CSA) to develop the opposing crow search algorithm (OCSA) for DoS attack traffic detection. This hybrid approach strategically leverages nature-inspired algorithms and machine learning to select optimal features for subsequent classification by an RNN classifier. Reddy and his team illustrate the innovative cross-pollination of methodologies, emphasizing the importance of interdisciplinary strategies in cybersecurity research.

Jinhui et al. [8], who introduced a power number correlation check method. This unique approach showcases the significance of hybrid solutions in the context of DoS attacks. Jinhui and his colleagues successfully demonstrated that their method not only improves the detection rate of malicious nodes but also effectively mitigates the impact of hybrid DoS attacks on network traffic.

J. Singh et al., [9] conducted a comprehensive study on various ways to counter Distributed Denial-of-Service (DDoS) attacks within the framework of software-defined networking (SDN). They carefully examined different methods to safeguard SDN systems, identifying weak points and categorizing DDoS attacks based on their impact. The authors thoroughly reviewed 75 influential articles that categorized DDoS defense solutions, considering factors like attack targets, defense tactics, testing settings, and traffic generation techniques. They also highlighted gaps and challenges in existing research, providing valuable direction for future investigations. This survey can guide the development of stronger DDoS defense systems for SDN networks, a pressing need given the expansion of technologies like NFV, IoT, and cloud computing, alongside the growing adoption of SDN in data centers.

F. Sales et al., [10] proposed a machine learning-driven approach to spot denial-of-service (DoS) attacks in networks. Their technique employs unique patterns taken from network traffic, achieving a remarkable detection rate (over 96%) using only a fraction of network data. Their work aligns with a broader survey of network security literature, confirming the prowess of machine learning-based methods in pinpointing DoS attacks effectively. These findings echo a growing trend of research exploring machine learning's potential in DoS/DDoS attack detection, indicating a promising direction for further exploration in the field.

T. Alharbi et al., [11] introduced a novel approach to detect DDoS attacks, a significant threat to network infrastructure. They emphasize the drawbacks of current detection systems including early detection challenges, computation intensity, and accuracy concerns. Their proposed solution employs deep inspection to differentiate genuine and attack traffic, utilizing NFV to classify diverse DDoS attack types. This approach cleverly addresses existing limitations, contributing to literature with an innovative strategy for

DDoS detection through NFV. The work extends prior network security research and introduces a fresh perspective to overcome prevailing constraints in current methodologies.

S. Haider et al., [12] introduced an innovative approach to detect DDoS attacks within software-defined networks, employing a deep CNN ensemble framework. Emphasizing the escalating prevalence and complexity of cyber threats, especially DDoS attacks, the authors contend that SDNs present a promising mitigation avenue. Their proposed framework capitalizes on the capabilities of CNNs and ensembles, enhancing accuracy. It undergoes evaluation using an advanced flow-based dataset, demonstrating superior precision compared to existing detection techniques.

Gadze et al., [13] introduced Software-Defined Networking (SDN), elucidating its control-data plane separation and centralized controllers, along with associated security challenges like DDoS attacks. Distributed Denial of Service (DDoS) attacks are defined, emphasizing their disruptive nature and challenges in SDN contexts. Deep learning and machine learning's role in network security are outlined, focusing on convolutional neural networks (CNNs) and Long Short-Term Memory (LSTM) networks for pattern recognition. The significance of model evaluation metrics, data splitting, and comparisons with classical machine learning models is discussed.

T. Abhiroop, et al., [14] discussed encompasses Denial of Service (DoS) attacks, highlighting their disruptive potential and the need for robust detection in SDN environments. The integration of machine learning for network security is introduced, emphasizing its role in analyzing network data and identifying malicious patterns. Feature extraction from switch-controller communication traces and flow-table snapshots is explained, along with the application of machine learning algorithms such as Neural Network, Support Vector Machines, and Naive Bayes for classification. The survey underlines the importance of evaluation metrics and the notable 100% accuracy achieved by the proposed approach.

Ye, et al., [15] explained the significance of detecting DDoS attacks for network security, outlining traditional detection methods. Introduction to Software Defined Networks (SDN) highlights their advantages in dynamic management. The survey explores the role of deep learning, like neural networks, in DDoS detection and acknowledges challenges in applying deep learning within SDN environments. It introduces Support Vector Machines (SVM) as a viable alternative in network security, emphasizing its effectiveness in binary classification tasks. The methodology section outlines the creation of a simulated SDN environment and the extraction of flow table values for SVM-based DDoS detection. Experimental results showcase the proposed method's impressive 95.24% accuracy rate in detecting DDoS attacks with minimal flow data.

SR NO	AUTHOR	ADVANTAGES	DISADVANTAGES
1	Vishwakarma et al. [1]	Explored DDoS attack mitigation in IoT and SDN systems using ML, providing insights into defense strategies.	Limited scope, focusing only on IoT and SDN domains, overlooking other potential DDoS attack vectors.
2	Tripathi et al. [2]	Categorized attacks based on application layer protocols (ALP) and divided attacks into protocol-specific and generic categories.	Study not exclusively focused on LDoS attacks.

3	M. S. Essayed et al. [3]	Systematic evaluation of machine learning methods for detecting malicious traffic in Software Defined Networks (SDNs).	Focus on benchmarking and fortified framework improves attack detection in SDNs.
4	Tang et al. [4]	The proposed SADBSCAN algorithm introduces density-based clustering to network traffic analysis. It effectively groups network data and employs cosine similarity for identifying DoS attacks in clusters.	While SADBSCAN is innovative in utilizing clustering and similarity metrics for attack detection, it might require fine-tuning parameters to adapt to different network environments. Implementation complexity could be a potential challenge for adoption in certain settings.
5	T. Alharbi et al., [5]	Introduces innovative DDoS defense approach using edge computing and NFV, addressing hardware limitations, privacy issues, and delays.	Departs from traditional security methods, potentially requiring a shift in infrastructure and expertise.
6	Osorio et al. [6]	Demonstrated efficacy of GMM and UBM in countering DoS/DDoS attacks, enhancing network security.	Potential complexity in implementing advanced statistical models and machine learning principles.
7	Reddy et al. [7]	Integration of OBL and CSA enhances DoS attack detection	Complexity of hybrid approach may require advanced implementation
8	Jinhui et al. [8]	Innovative hybrid approach enhances malicious node detection and mitigates hybrid DoS impact.	Method details may be complex.
9	J. Singh et al. [9]	Thoroughly examines DDoS defense in SDN, categorizing attacks and providing future guidance.	Limited focus on potential emerging technologies.
10	F. Sales et al., [10]	Introduced machine learning-based approach for DoS attack detection. Achieved high detection rate (over 96%) using minimal network data. Confirms effectiveness of ML methods in identifying DoS attacks	Specific technique details not outlined, limiting insight into the exact methods used. Lack of discussion on potential challenges or limitations.
11	T. Alharbi et al.,	Innovative NFV-based DDoS detection strategy	Challenges in early detection and Potential complexity in implementing NFV
12	S. Haider et al., [12]	Innovative DDoS detection approach in software-defined networks (SDNs).	Complex implementation due to deep CNN ensemble framework.
13	Gadze et al. [13]	Clarifies the core elements of SDN, emphasizing control-data plane division and centralized controllers.	The focus on SDN-specific challenges might limit broader insights into network security issues.
14	T. Abhiroop, et al. [14]	Discusses Denial of Service (DoS) attacks' disruptive potential, emphasizing robust detection in SDN environments. Introduces machine learning for network security, highlighting its role in analyzing network data and identifying malicious patterns.	While the survey underlines the importance of evaluation metrics, it lacks insight into potential limitations or challenges faced by the proposed approach.
15	Ye, et al., [15]	Highlights DDoS attack importance and traditional detection methods. Explores SDN and deep learning role. Introduces SVM efficacy in network security.	Acknowledges challenges in deep learning application within SDN. Experimental accuracy relies on minimal flow data.

Research Gap

The research gaps in DDoS attack detection encompass the need for comprehensive defense frameworks across diverse networks, adaptive mechanisms, emerging attack strategies, privacy-preserving methods, hybrid defenses, zero-day resilience, and resource-efficient algorithms. Existing studies often focus on known attacks, lacking scalability validation. Bridging these gaps can lead to innovative, adaptable, and privacy-respecting solutions to counter evolving DDoS threats.

3. Methodology

3.1 Proposed system

The Proposed system is designed to tackle the increasing danger of Denial of Service (DoS) attacks. By using advanced deep learning methods, the system aims to categorize and recognize various DoS attacks, making it possible to find and stop them effectively. The system uses CLSTM for sorting out the attacks, as shown in Figure 5. Additionally, the SVM Algorithm is used to compare the results.

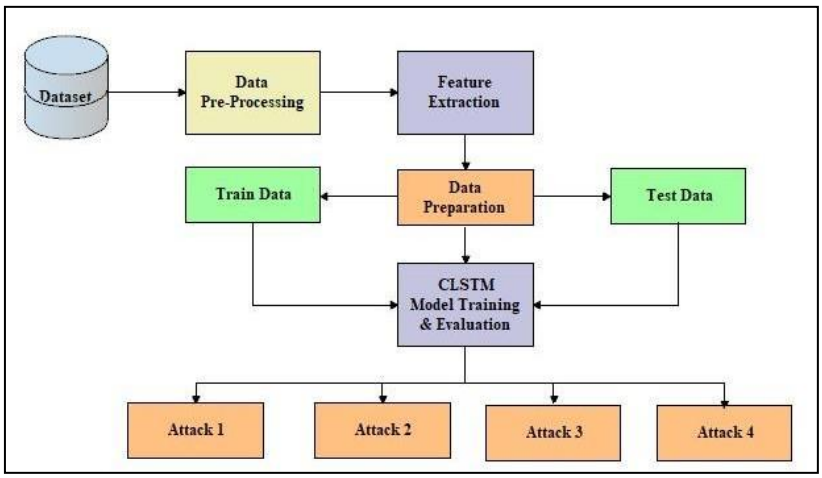


Fig 2: The Proposed System

- **Data Collection:** The initial step involves gathering relevant data pertaining to DoS attacks. This data serves as the foundation for subsequent analysis and model development.
- **Data Preprocessing:** Once the data is collected, it undergoes preprocessing to enhance its quality and consistency. This includes tasks like handling missing values, data normalization.
- **Feature Extraction:** Extracting meaningful features from the preprocessed data is crucial for accurate analysis. This stage involves selecting and transforming relevant attributes that contribute to identifying DoS attacks.
- **Data Splitting:** The dataset is divided into training and testing subsets. The training subset is used to teach the model to recognize patterns, while the testing subset evaluates the model's performance on new, unseen data.
- **CLSTM Model Training and Evaluation:** The heart of the proposed system lies in training a Convolutional Long Short-Term Memory (CLSTM) neural network. This advanced deep learning model is specifically designed for sequence-based data analysis, making it suitable for detecting DoS

attacks. The model is trained using the training data and evaluated using the testing data to ensure its effectiveness.

3.2 Research Methodology

Data mining projects are typically created using the CRISP-DM (Cross-Industry Standard Procedure for Data Mining) technique, which involves six distinct phases.

1. Business Understanding: In the project's initial phase, understanding objectives, stakeholders, and scope is crucial. Stakeholder interviews, data analysis, and risk assessment ensure clear goals, risk management, and business-aligned success.

2. Understanding the Data: Data exploration involves gathering and analyzing attributes to understand their utility. It guides the project by assessing quality, identifying trends, and addressing concerns through descriptive statistics and visualizations. Quality assurance is essential for reliability and decision-making, setting a foundation for subsequent project stages.

Dataset details: The dataset comprises 117,460 rows and 78 columns, featuring four distinct labels: Benign, MSSQL, Syn, and UDP.

Dataset Source: *https://www.kaggle.com/datasets/dhoogla/cicddos2019.*

3. Preparation of the Data: During this phase, data undergoes meticulous cleaning, transformation, and preparation, addressing missing values, refining features, and utilizing sampling techniques. It ensures structured, high-quality data for effective modeling, forming the foundation for successful analysis and modeling-stages.

4. Implementation

4.1 Label Encoding

```
class_dict={}
for idx,label in enumerate(class_labels):
    class_dict[label]=idx
print(class_dict)
```

The above code snippet in Python creates a dictionary named **class_dict** that establishes a mapping between class labels and their respective indices. Through a loop, each class label is paired with an index value using the enumerate function. The loop iterates over a list of **class_labels,** assigning each label an index within the dictionary. This mapping is crucial for tasks like machine learning, where numerical representations of class labels are required. Once the loop is completed, the resulting **class_dict** dictionary holds this mapping information.

4.2 Feature Selection

```
target_feature = 'class'
all_features = df.columns.tolist()
all_features.remove(target_feature)
corr = df[all_features].corrwith(df[target_feature])
sorted_features = corr.abs().sort_values(ascending=False).index.tolist()
selected_features = sorted_features[:20]
filtered_df = df[[target_feature] + selected_features]
```

The above code snippet performs a feature selection process based on correlation analysis within a given **DataFrame (df).** The objective is to identify and isolate the most influential features that exhibit a strong correlation with a specific target feature, marked as 'class'.

The process begins by defining the target feature as 'class'. Subsequently, all the features within the **DataFrame** are enumerated and stored in the **all_features** list. The target feature is then removed from this list to prepare for correlation calculations.

The code proceeds to compute the correlation between each feature in the **all_features** list and the target feature ('class'). The resultant correlation coefficients are stored in the **corr** Series.

To identify the features with the highest correlation, the absolute values of the correlation coefficients are sorted in descending order. The corresponding feature names are extracted, generating the **sorted_features** list. This list ranks the features based on their correlation strength with the target feature.

The next step involves selecting the top 20 features from the sorted list, which have the highest absolute correlation values. These selected features are stored in the **selected_features** list.

Finally, a new **DataFrame, filtered_df,** is created by extracting the 'class' target feature and the top 20 selected features. This subset of the original DataFrame retains only the most influential features for further analysis or modeling. The code employs correlation analysis to identify and retain the most relevant features, streamlining the data and enhancing its suitability for subsequent analysis tasks.

4.3 Normalization

```
scaler=MinMaxScaler()
scaler=scaler.fit(filtered_df.drop(labels='class',axis=1))
scaled_df=scaler.transform(filtered_df.drop(labels='class',axis=1))
data=pd.DataFrame(data=scaled_df,columns=filtered_df.drop(labels='class',axis=1).columns)
data['class']=filtered_df['class'].values
data.head()
```

The provided code snippet conducts data preprocessing through Min-Max Scaling on a DataFrame named **filtered_df**. This scaling technique standardizes the ranges of features to a uniform interval, aiding various analytical and modeling processes. The code first initializes a MinMaxScaler instance. Next, the **scaler** is fitted to the data after excluding the 'class' column, enabling computation of feature scaling parameters. The data is then transformed using these parameters, and the scaled result is stored in the **scaled_df** array.

A new DataFrame named data is created from the scaled data, inheriting column names from the original **filtered_df** while excluding the 'class' column. The 'class' column is subsequently appended back to the data DataFrame, ensuring preservation of class labels.

This preprocessing enhances the uniformity of feature scales and prepares the data for subsequent analyses. Min-Max Scaling supports improved performance of algorithms sensitive to feature magnitudes. By showcasing the initial rows of the preprocessed data DataFrame, the snippet offers a quick view of the transformed data.

4.4 Data Oversampling

```
from imblearn.over_sampling import SMOTE
oversample = SMOTE()
X_smote, y_smote = oversample.fit_resample(X.values, y.values.ravel())

data=pd.DataFrame(data=X_smote,columns=X.columns)
data['class']=y_smote
data=data.sample(frac=1).reset_index(drop=True)
data.head()
```

The above code snippet employs the Synthetic Minority Over-sampling Technique (SMOTE) to tackle class imbalance in a dataset. First, the SMOTE class from the imbalanced-learn library is imported. An instance of SMOTE is created, and it is applied to the dataset containing features (X) and corresponding class labels (y).

SMOTE generates synthetic samples for the minority class, effectively oversampling it to balance class distribution. The oversampled data is stored in arrays **X_smote** and **y_smote.**

A new DataFrame called data is constructed using the oversampled feature matrix **(X_smote).** Columns are named based on the original features, and the target vector **(y_smote)** is added as a 'class' column.

To ensure randomness, the data in the data DataFrame is shuffled. The code then prints the initial rows of the shuffled and oversampled data DataFrame, providing an overview of the processed dataset.

4. Modeling: Data is used to develop and assess predictive models, involving algorithm selection, training, testing, and evaluation. Algorithms like decision trees, logistic regression, linear regression, and support vector machines are commonly chosen based on data and goals. Data partitioning improves generalization, while precision and accuracy metrics gauge model performance.

4.5 Model: CLSTM

```
model = Sequential()
model.add(Input(shape=(x_train.shape[1], x_train.shape[2])))
model.add(Conv1D(filters=64,kernel_size=3,activation='relu',padding='same'))
model.add(Conv1D(filters=64,kernel_size=3,activation='relu',padding='same'))
model.add(MaxPool1D(pool_size=2))
model.add(Conv1D(filters=128,kernel_size=3,activation='relu',padding='same'))
model.add(Conv1D(filters=128,kernel_size=3,activation='relu',padding='same'))
model.add(MaxPool1D())
model.add(Conv1D(filters=256,kernel_size=3,activation='relu',padding='same'))
model.add(Conv1D(filters=256,kernel_size=3,activation='relu',padding='same'))
model.add(MaxPool1D())
model.add(Conv1D(filters=512,kernel_size=3,activation='relu',padding='same'))
model.add(Conv1D(filters=512,kernel_size=3,activation='relu',padding='same'))
model.add(BatchNormalization())
model.add(MaxPool1D())
model.add(Bidirectional(LSTM(units=200,return_sequences=True,recurrent_dropout=0,activation='tanh',recurrent_activation='sigmoid',unroll=False,use_bias=True)))
model.add(Bidirectional(LSTM(units=200,return_sequences=True,recurrent_dropout=0,activation='tanh',recurrent_activation='sigmoid',unroll=False,use_bias=True)))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(512, activation='relu'))
model.add(Dense(4, activation='softmax'))
model.compile(loss='categorical_crossentropy',optimizer="adam",metrics=['accuracy'])
```

The above code snippet outlines the construction and compilation of a deep learning model using the Keras library with TensorFlow as its backend. The model architecture is designed for sequential data processing, making it suitable for tasks involving time series analysis or sequential data classification.

- The core elements of the code are as follows:
- The model is initialized as a sequential container, forming a linear stack of layers.
- An input layer is added, specifying the input shape based on the dimensions of the training data **(x_train).**
- A series of Conv1D layers is sequentially appended, employing one-dimensional convolution to extract key features from the input data. Parameters like filters, **kernel_size,** activation, and padding control their behavior.
- MaxPool1D layers are utilized for down-sampling, reducing data dimensionality while retaining crucial features.
- Additional Conv1D and MaxPool1D layers are integrated to further process extracted features.
- BatchNormalization() ensures stable activations by normalizing the previous layer's output.
- Bidirectional LSTM layers are incorporated to capture temporal dependencies from both directions, essential for sequential data analysis. Parameters like units, **return_sequences**, and **recurrent_dropout** impact their functionality.
- A Flatten() layer transforms LSTM outputs into a flat vector.

- Regularization is introduced via Dropout(0.5) to mitigate overfitting.
- Fully connected Dense layers are employed for classification. The final layer with softmax activation facilitates multi-class classification, with 4 output units.
- The model.compile() function configures the model for training, specifying loss function, optimizer, and evaluation metrics.
- A summary of the model's architecture is displayed via model.summary(), providing insights into the layers' parameters and data flow.

4.6 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a powerful and versatile machine learning algorithm that has found widespread application in various domains due to its effectiveness in both classification and regression tasks. Introduced by Vapnik and his colleagues in the 1990s, SVM has since become a cornerstone of modern machine learning.

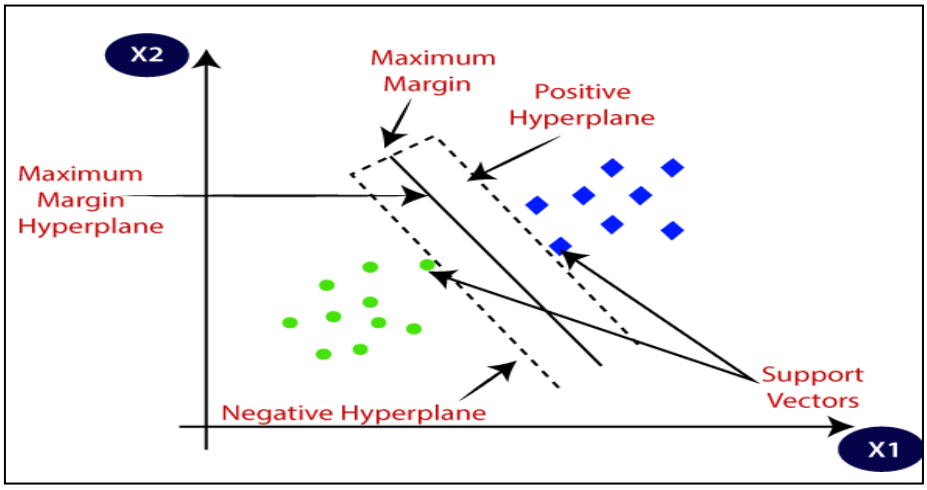


Fig 3: The SVM Architecture[3]

At its core, the Support Vector Machine (SVM) is a supervised learning technique designed to discover a hyperplane within a high-dimensional feature space that effectively segregates distinct data point classes. This hyperplane is strategically positioned to maximize the margin between classes, creating a distinct boundary. This exceptional approach not only enhances accurate classification but also promotes robust generalization to new, unseen data. SVM is versatile, handling both linear and non-linear classifications through the kernel trick. Kernels enable SVM to implicitly transform data into higher-dimensional spaces, effectively distinguishing non-linearly separable classes. This adaptability empowers SVM to address intricate data distributions and capture complex decision boundaries. A unique feature of SVM is its emphasis on support vectors, the data points near the decision boundary. These vectors significantly contribute to defining the optimal hyperplane, rendering SVM highly resilient to outliers. By prioritizing

[3] https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm

these support vectors, SVM achieves remarkable generalization performance across a spectrum of data scenarios.

4.7 CLSTM

Convolutional Long Short-Term Memory (CLSTM) is a sophisticated neural network architecture that merges the strengths of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. The purpose of CLSTM is to capitalize on the unique capabilities of both CNNs and LSTMs, making it an ideal choice for tasks involving sequential data, such as time series analysis or sequential image processing.

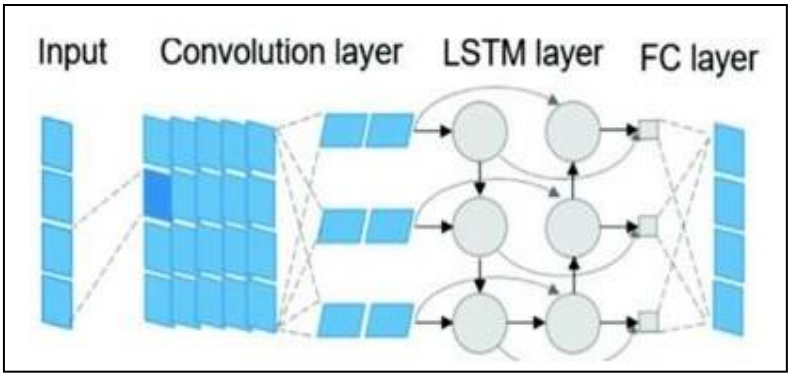


Fig 4: The CLSTM Architecture **[16]**

Convolutional Neural Networks (CNNs) have surged in popularity within computer vision, particularly for image-related tasks. Their success lies in their capacity to autonomously learn significant features from raw image data. Comprising layers including convolutional and pooling layers, CNNs employ filters to extract spatial features, and pooling layers down sample, preserving crucial data. In contrast, Long Short-Term Memory (LSTM) Networks are part of the Recurrent Neural Network (RNN) family, adept at handling sequential data with temporal relationships. LSTMs excel in capturing context over extended sequences through specialized memory cells, evading the vanishing gradient challenge. By uniting CNNs and LSTMs in the CLSTM architecture, a versatile model emerges. CNN layers excel at spatial feature extraction, while LSTM layers leverage memory cells to model temporal dependencies. This fusion empowers CLSTM to grasp both spatial and temporal aspects, fitting tasks involving sequential data like time series analysis or sequential image processing. CLSTM boasts broad applications, particularly in forecasting domains such as finance and weather prediction, where it adeptly anticipates future values based on past observations.

5. Results Analysis

5.1 Support Vector Machine

5.1.1 Support Vector Machine - Report on Classification

A classification report is in fact a performance review metric in machine learning. It is used to show the accuracy, memory, F1 Measure, and support of the trained classification model. For the sake of clarity, Fig.6 provides each of the indicators from the Support Vector Machine Classifier classification report.

	precision	recall	f1-score	support
Benign	0.99	0.92	0.96	9768
MSSQL	1.00	0.10	0.19	9768
Syn	0.93	0.99	0.96	9768
UDP	0.53	1.00	0.69	9768
accuracy			0.75	39072
macro avg	0.86	0.75	0.70	39072
weighted avg	0.86	0.75	0.70	39072

Tab 1: Classification Report for Support Vector Machine Classifier

The given table presents a summary of classification performance metrics for different classes. For the "Benign" class, precision is 0.99, recall is 0.92, and the F1-score is 0.96, with a support of 9768 instances. The "MSSQL" class has a precision of 1.00, recall of 0.10, and F1-score of 0.19, also with a support of 9768 instances. The "Syn" class exhibits a precision of 0.93, recall of 0.99, and F1-score of 0.96, with support again at 9768 instances. The "UDP" class demonstrates a precision of 0.53, recall of 1.00, and F1-score of 0.69, with 9768 instances of support. The overall accuracy across all classes is 0.75. The "Macro Avg" values for precision, recall, and F1-score are 0.86, 0.75, and 0.70, respectively, with a total support of 39072 instances. Similarly, the "Weighted Avg" metrics are also 0.86, 0.75, and 0.70, with the same support of 39072 instances. This comprehensive table offers insights into the model's classification performance for each class and on average.

5.1.2 Support Vector Machine Classifier - Confusion Matrix

A confusion matrix, which presents the many situations of the prediction & findings in a tabular format, helps with visualizing the outcomes of a classification exercise. Given in below fig. It generates a table containing all of the predicted & actual values from a classifier.

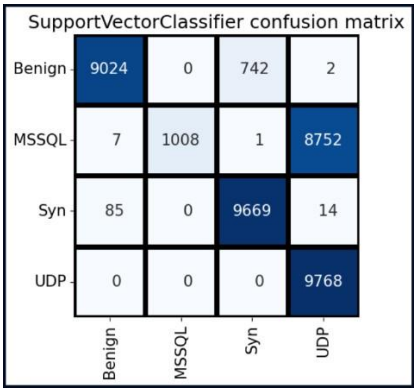


Fig 5: Confusion Matrix for Support Vector Machine Classifier

The given confusion matrix provides valuable insights into a classification model's performance across four target classes: "Benign," "MSSQL," "syn," and "UDP." The matrix breakdown based on the provided information is as follows:

For the "Benign" Class:

- True Positives (TP): 9024 (Instances correctly classified as "Benign" by the model)
- False Positives (FP): 744 (Instances incorrectly labeled as "Benign" when they actually belong to the "DDOS" class)

For the "MSSQL" Class:

- True Positives (TP): 8752 (Instances correctly classified as "MSSQL" by the model)
- False Positives (FP): 1016 (Instances incorrectly labeled as "MSSQL" when they belong to the "Benign" class)

For the "syn" Class:

- True Positives (TP): 9669 (Instances correctly classified as "syn" by the model)
- False Positives (FP): 99 (Instances incorrectly labeled as "syn" when they belong to the "DDOS" class)

For the "UDP" Class:

- True Positives (TP): 9768 (Instances correctly classified as "UDP" by the model)
- False Positives (FP): 0 (Instances incorrectly labeled as "UDP" when they belong to the "DDOS" class)

This comprehensive breakdown of the confusion matrix elucidates the model's ability to accurately classify instances into the specified target classes. It showcases both correct predictions (true positives) and instances where the model made incorrect classifications (false positives), offering a clear assessment of the model's strengths and areas for improvement in correctly categorizing instances within these four classes.

5.2 Convolutional Long Short-Term Memory

5.2.1 Convolutional Long Short-Term Memory - Classification Report

A classification report serves as a performance evaluation metric in machine learning. It is used to show the precision, memory, support, and F1 Measure of the trained classification model. For the sake of clarity, displays in below fig each signal from its Convolutional Long Short-Term Memory classification report.

	precision	recall	f1-score	support
Benign	0.99	1.00	1.00	9768
MSSQL	0.97	1.00	0.98	9768
Syn	1.00	0.99	1.00	9768
UDP	1.00	0.97	0.98	9768
accuracy			0.99	39072
macro avg	0.99	0.99	0.99	39072
weighted avg	0.99	0.99	0.99	39072

Tab 2: Classification Report for Convolutional Long Short-Term Memory

The presented table provides a comprehensive overview of classification performance metrics for distinct classes. The metrics encompass precision, recall, F1-score, and support, each shedding light on the model's effectiveness. Specifically, for the "Benign" class, the precision reaches 0.99, indicating highly accurate positive predictions, while recall at 1.00 underscores the model's ability to capture the majority of actual "Benign" instances. This class also attains an F1-score of 1.00, signifying a harmonious balance between precision and recall. The "MSSQL" class demonstrates a precision of 0.97, implying accurate positive predictions, though with a recall of 1.00, suggesting some instances might have been missed. The "Syn" class achieves an impressive F1-score of 1.00, backed by precision and recall values at 1.00 and 0.99, respectively. The "UDP" class showcases a balanced performance with precision, recall, and F1-score all hovering around 0.97 to 1.00. These combined metrics ultimately contribute to an overall accuracy of 0.99 across the entire dataset. Additionally, the macro and weighted averages, both at 0.99, indicate a consistent performance across classes.

5.2.2 Convolutional Long Short-Term Memory - Confusion Matrix

A confusion matrix helps visualize the results of a classification task by providing a tabular arrangement of different scenarios of predictions and detections. This is shown below in table with values.

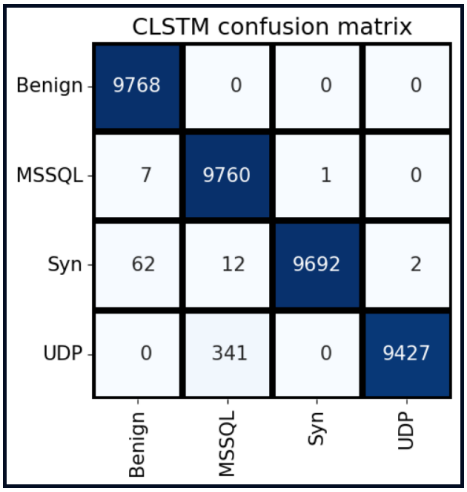


Fig 6: Confusion Matrix for Convolutional Long Short-Term Memory

The provided confusion matrix offers significant insights into the classification model's performance across four distinct target classes: "Benign," "MSSQL," "syn," and "UDP." The matrix's detailed breakdown using the provided data is as follows:

For the "Benign" Class:

- True Positives (TP): 9768 (Instances correctly classified as "Benign" by the model)
- False Positives (FP): 0 (Instances incorrectly labeled as "Benign" but belong to the "DDOS" class)

For the "MSSQL" Class:

- True Positives (TP): 9760 (Instances correctly classified as "MSSQL" by the model)
- False Positives (FP): 8 (Instances incorrectly labeled as "MSSQL" but belong to the "Benign" class)

For the "syn" Class:

- True Positives (TP): 9692 (Instances correctly classified as "syn" by the model)
- False Positives (FP): 76 (Instances incorrectly labeled as "syn" but belong to the "DDOS" class)

For the "UDP" Class:

- True Positives (TP): 9427 (Instances correctly classified as "UDP" by the model)
- False Positives (FP): 341 (Instances incorrectly labeled as "UDP" but belong to the "DDOS" class)

This comprehensive depiction of the confusion matrix underscores the model's efficacy in accurately categorizing instances within the specified target classes. The matrix reveals both accurate predictions (true positives) and instances where the model made incorrect classifications (false positives), providing a lucid evaluation of the model's strengths and areas for potential enhancement in precise classification across these four classes.

5.3 Accuracy Comparison Graph

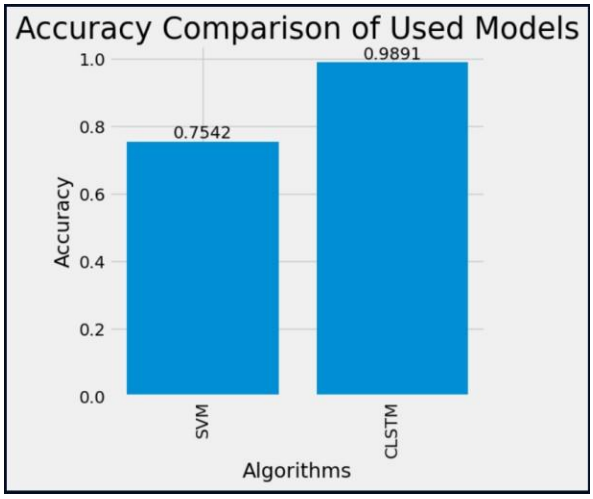


Fig 7: Accuracy Comparison Models graphs

The accuracy comparison between models demonstrates a notable contrast. The Support Vector Machine (SVM) classifier achieved an accuracy of 75.42%, showcasing its capability to discern patterns in data. On the other hand, the Convolutional Long Short-Term Memory (CLSTM) classifier exhibited a significantly higher accuracy of 98.91%, indicating its proficiency in capturing intricate temporal dependencies within sequences. This comparison underscores the superior performance of the CLSTM model in this scenario. It's imperative to consider the specific task's requirements and dataset characteristics while selecting between the two models, as the CLSTM's enhanced accuracy could be pivotal for tasks demanding precise sequence-based classifications.

6. Evaluation:

The evaluation of the CLSTM algorithm involves a structured process to assess its efficacy in detecting intrusions. The study utilizes a set of 20 diverse test files, providing a representative sample of potential attack scenarios. Each test file represents a unique intrusion attempt. By selecting any test file and subjecting it to the CLSTM algorithm, its performance in accurately identifying the type of intrusion is analyzed. The algorithm's results are then linked to a designated email alert system, promptly notifying relevant personnel about detected intrusions. This comprehensive evaluation not only validates the algorithm's detection capabilities but also assesses its real-world applicability, thereby contributing to advancements in intrusion detection for the field of cybersecurity.

6.1 Test Case – 1

```
input_=pd.read_csv("user_input/file_10.csv")
input_.head()
```

In this test case, I have got the input from **file_10.csv** and after running the model we will get the notification of the intrusion and its type on the mail and as we can see I got mail for **SYN Intrusion.**

6.2 Test Case – 2

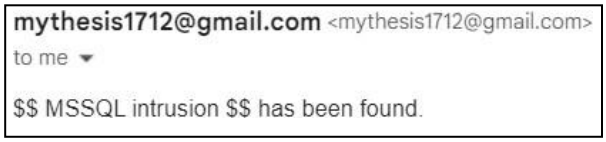
```
input_=pd.read_csv("user_input/file_16.csv")
input_.head()
```

In this test case, I have got the input from **file_16.csv** and after running the model we will get the notification of the intrusion and its type on the mail and as we can see I got mail for **UDP Intrusion.**

6.3 Test Case – 3

```
input_=pd.read_csv("user_input/file_7.csv")
input_.head()
```

In this test case, I have got the input from **file_7.csv** and after running the model we will get the notification of the intrusion and its type on the mail and as we can see I got mail for **MSSQL Intrusion.**



7. Conclusion and Future Work:

The rising threat of Denial-of-Service (DOS) attacks in cybersecurity necessitates innovative solutions. This study centers on countering DOS attacks using advanced Deep Learning techniques, with the Convolutional Long Short-Term Memory (CLSTM) neural network leading as a primary tool for classification. Meticulous analysis of network traffic data categorizes activities into classes, and the CLSTM model achieves an impressive 98.91% detection accuracy, outperforming the SVM model's 75.42%. Looking ahead, integrating CLSTM with attention mechanisms or transformers holds promise, along with real-time implementation, anomaly detection, data augmentation, ensemble methods, and transfer learning. Dynamic adaptation, interpretability, scalability, and establishing predictive-network adjustment feedback loops are essential. These pursuits aim to bolster network resilience against evolving threats, advancing the field of cybersecurity

References

1. R. Vishwakarma and A. K. Jain, ‘‘A survey of DDoS attacking techniques and defence mechanisms in the IoT network,’’ Telecommun. Syst., vol. 73, no. 1, pp. 3–25, Jul. 2019.
2. N. Tripathi and N. Hubballi, ‘‘Application layer denial-of-service attacks and defense mechanisms: A survey,’’ ACM Comput. Surveys, vol. 54, no. 4, pp. 1–33, May 2022, doi: 10.1145/3448291.
3. M. S. Elsayed, S. Dev, and A. D. Jurcut, “MachineLearning Techniques for Detecting Attacks in SDN,” arXiv preprint arXiv: 1910.00817, 2019.
4. D. Tang, S. Zhang, J. Chen, and X. Wang, “&e detection of low-rate DoS attacks using the SADBSCAN algorithm,” Information Sciences, vol. 565, pp. 229–247, 2021
5. T. Alharbi, A. Aljuhani, and H. Liu, ‘‘Holistic DDoS mitigation using NFV,’’ in Proc. IEEE 7th Annu. Comput. Commun. Workshop Conf. (CCWC), Las Vegas, NV, USA, Jan. 2017, pp. 1–4, doi: 10.1109/CCWC.2017.7868480.
6. J. S. M. Osorio, J. A. V. Tejada, and J. F. B. Vega, “Detection of DoS/DDoS attacks: the UBM and GMM approach,” in Proceedings of the 2021 IFIP/IEEE International Symposium on Integrated Network Management (IM), pp. 866–871, IEEE, France, May 2021.
7. S. S. T. Reddy and G. K. Shyam, “An efficient metaheuristic algorithm based feature selection and recurrent neural network for DoS attack detection in cloud computing environment,” Applied Soft Computing, vol. 100, Article ID 106997, 2021.
8. X. Jinhui, T. Yang, and Y. Feiyue, “Intrusion detection system for hybrid DoS attacks using energy trust in wireless sensor networks,” Procedia Computer Science, vol. 131, pp. 1188– 1195, 2018.
9. J. Singh and S. Behal, “Detection and mitigation of DDoS attacks in SDN : A comprehensive review , research challenges and future directions,” Computer Science Review, vol. 37, p. 100279, 2020.
10. F. Sales et al., “Smart Detection: An Online Approach for DoS / DDoS Attack Detection Using Machine Learning,” Security and Communication Networks, vol. 2019, 2019.
11. T. Alharbi, A. Aljuhani, H. Liu, and C. Hu, ‘‘Smart and lightweight DDoS detection using NFV,’’ in Proc. Int. Conf. Compute Data Anal. (ICCDA), Lakeland, FL, USA, 2017, pp. 220–227.
12. S. Haider et al., “A Deep CNN Ensemble Framework for Efficient DDoS Attack Detection in Software Defined Networks,” IEEE Access, vol. 8, pp. 53972– 53983, 2020.
13. Gadze, J.D.; Bamfo-Asante, A.A.; Agyemang, J.O.; Nunoo-Mensah, H.; Opare, K.A.-B. An Investigation into the Application of Deep Learning in the Detection and Mitigation of DDOS Attack on SDN Controllers. *Technologies* 2021, *9*, 14. https://doi.org/10.3390/technologies9010014
14. T. Abhiroop, S. Babu and B. S. Manoj, "A Machine Learning Approach for Detecting DoS Attacks in SDN Switches," 2018 Twenty Fourth National Conference on Communications (NCC), Hyderabad, India, 2018, pp. 1-6, doi: 10.1109/NCC.2018.8600196.
15. Ye, J.; Cheng, X.; Zhu, J.; Feng, L.; Song, L. A DDoS Attack Detection Method Based on SVM in Software Defined Network. Secur. Commun. Netw. 2018, 2018, 1–8.
16. Kim, Tae-Young, Cho, Sung-Bae, "Predicting Residential Energy Consumption using CNN-LSTM Neural Networks," Energy, Elsevier, vol. 182.C (2019), 72-81