

# Enhancing SDN security with a feature-based approach using multiple k-means, Word2Vec, and neural network

Hicham Yzzogh, Hafssa Benaboud

Intelligent Processing Systems and Security (IPSS), Faculty of Sciences, Mohammed V University in Rabat, Rabat, Morocco

## Article Info

### Article history:

Received Jun 9, 2024

Revised Oct 15, 2024

Accepted Nov 19, 2024

### Keywords:

Classification algorithms

Clustering algorithms

K-means

Neural network

Software-defined networking

Software-defined networking security

Word2Vec

## ABSTRACT

In the rapidly evolving landscape of network management, software-defined networking (SDN) stands out as a transformative technology. It revolutionizes network management by decoupling the control and data planes, enhancing both flexibility and operational efficiency. However, this separation introduces significant security challenges, such as data interception, manipulation, and unauthorized access. To address these issues, this paper investigates the application of advanced clustering and classification algorithms for anomaly detection and traffic analysis in SDN environments. We present a novel approach that integrates multiple k-means clustering models with Word2Vec for feature extraction, followed by classification using a neural network (NN). Our method is rigorously benchmarked against a traditional NN model to comprehensively evaluate performance. Experimental results indicate that our approach outperforms the NN model, achieving an accuracy of 99.97% on the InSDN dataset and 98.65% on the CIC-DDoS2019 dataset, showcasing its effectiveness in detecting anomalies without relying on feature selection. These findings suggest that integrating clustering techniques with feature extraction algorithms can significantly enhance the security of SDN infrastructures.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



## Corresponding Author:

Hicham Yzzogh

Intelligent Processing Systems and Security (IPSS), Faculty of Sciences

Mohammed V University in Rabat

Avenue Ibn Battouta B.P. 1014 RP, Rabat, Morocco

Email: Hicham\_yzzogh@um5.ac.ma

## 1. INTRODUCTION

Software-defined networking (SDN) (RFC7426 [1] and RFC7276 [2]) has revolutionized network management by decoupling the control plane from the data plane, offering enhanced flexibility and scalability in network operations. However, this shift has introduced new security challenges that traditional networking approaches are ill-equipped to address. The centralized control plane in SDNs becomes a critical point of vulnerability, susceptible to unauthorized access, control plane attacks, and data breaches. Additionally, the abstraction of network intelligence from hardware introduces risks such as data interception and manipulation.

To address these challenges, advanced algorithms for clustering and classification are emerging as promising solutions. Clustering algorithms like k-means are useful for behavioral profiling, traffic segmentation, and anomaly detection within SDN networks. They enable the identification of normal behavior patterns and deviations, serving as an early warning system against potential threats. Classification algorithms, such as decision trees (DT) and neural networks (NNs), are crucial for traffic analysis and security policy enforcement. They enhance the ability to implement targeted security measures by distinguishing between legitimate and malicious activities.

Recent studies have significantly advanced SDN security through various clustering and classification algorithms for anomaly detection and traffic analysis. For instance, Zheng *et al.* [3] improve packet classification efficiency in SDNs with the range supported bit vector (RSBV) algorithm. Tan *et al.* [4] increase DDoS detection accuracy by combining k-means with k-nearest neighbors (KNN). Xu *et al.* [5] enhance detection efficiency using k-means++ and fast k-nearest neighbors (K-FKNN). In feature-based approaches, Jafarian *et al.* [6] integrates the NetFlow protocol with feature selection and C-support vector classification, achieving respectable accuracy, while Garg *et al.* [7] use an enhanced restricted Boltzmann machine (RBM) and a gradient descent-based support vector machine (SVM) for detecting suspicious flows. Deep learning methods are also explored. Tang *et al.* [8] employs a fully connected deep neural network (DNN) in conjunction with gated recurrent unit-recurrent neural network (GRU-RNN) on the NSL-KDD dataset [9] to identify abnormal activities in SDN networks. Meanwhile, Shaji *et al.* [10] introduce deep-discovery IDS, which uses multi-layer perceptrons (MLP) and feedforward (FF) ANN for attack detection in SDN, achieving 98.81% accuracy. Additionally, Staden and Brown [11] evaluate random forest (RF), KNN, and DT for traffic classification in SDN environments within the context of the internet of things (IoT), with RF demonstrating notable performance.

Despite these advancements, a gap remains in effectively integrating feature-based approaches that combine clustering techniques with feature extraction algorithms. For instance, Tan *et al.* [4] improves accuracy using k-means with KNN, and Xu *et al.* [5] achieves better results with k-means++ and K-FKNN. However, neither study explores the integration of k-means or k-means++ with feature extraction techniques. Additionally, Jafarian *et al.* [6] achieves respectable accuracy through feature selection but does not employ clustering or feature extraction methods.

Our work addresses this gap by proposing a novel method that integrates multiple k-means clustering models with Word2Vec for feature extraction, and then employs a NN for classification. Unlike traditional methods that cluster the entire dataset collectively [12], [13], our approach analyzes individual features independently using k-means, providing a more detailed representation of the data. We evaluate our proposed approach alongside a baseline NN model using the InSDN dataset [14], [15], demonstrating that our method achieves high detection accuracy. Additionally, we assess its performance on the CIC-DDoS2019 dataset [16], highlighting its high accuracy in distinguishing between normal traffic and various types of DDoS attacks.

This paper is structured as follows: section 2 presents the datasets used and describes the proposed approach. Section 3 evaluates and discusses the experimental results. Finally, section 4 concludes the paper.

## 2. METHOD

In this section, we present a novel approach for traffic analysis that leverages k-means clustering, Word2Vec, and NN models. This method aims to enhance the accuracy of anomaly detection and traffic classification within SDN environments. Additionally, we will detail the datasets used in our experiments, including the InSDN dataset, which focuses on SDN attacks, and the CIC-DDoS2019 dataset for multi-class traffic classification.

### 2.1. Datasets

In our research, we use the InSDN dataset, which focuses on SDN attacks, to test our model. This dataset is offering a comprehensive collection of attack scenarios specifically designed for SDN networks. It's crucial to use an SDN-specific dataset when evaluating SDN attack detection methods, as generic datasets may not accurately reflect the unique architecture and attack vectors of SDN networks.

The InSDN dataset is available in .PCAP and .CSV formats and is divided into three groups: Normal, OVS, and Metasploitable-2. The Normal group represents typical user traffic, while the OVS group simulates various attacks, including brute force attacks (BFAs), BotNet attacks, denial-of-service (DoS), DDoS, Probes, and Web Attacks. To ensure a comprehensive representation of both regular network traffic and various types of attacks, we merge the Normal and OVS datasets in .CSV format.

Given that the BotNet and Web-Attack classes comprise 164 and 192 instances respectively in the merged dataset, we chose to exclude these classes from our experimental data. Figure 1 illustrates the distribution of instances for each class in our experimental dataset. Predominant occurrences are observed in Normal, DDoS, DoS, and Probe classes, while BFA contains only 1,110 instances. We retain only the relevant features and manually discard the irrelevant ones. Features such as 'Flow ID' and 'Timestamp' are discarded as they do not add any valuable information for addressing the problem. Additionally, we omit 'Src Port', 'Dst IP', and 'Dst Port' to prevent potential overfitting of the model. Consequently, our experimentation is conducted using the dataset identified as InSDN traffic analysis [17].

To further evaluate our proposed model, we utilize the CIC-DDoS2019 dataset. We create our experimental dataset with the distribution of instances illustrated in Figure 2. Our analysis focuses on

classifying traffic, encompassing five types of DDoS attacks as well as benign traffic. We manually remove the following features: 'Unnamed: 0', 'Flow ID', 'Timestamp', 'Source IP', 'Source Port', 'Destination IP', and 'Destination Port'. Therefore, we conduct our experiments using the dataset labeled CIC-DDoS2019 traffic analysis [18]. Both datasets used in our experiments are split into two subsets: 80% for training and 20% for testing. The testing set is utilized to assess the model's performance on entirely new, unseen data, providing a reliable estimate of its real-world effectiveness.

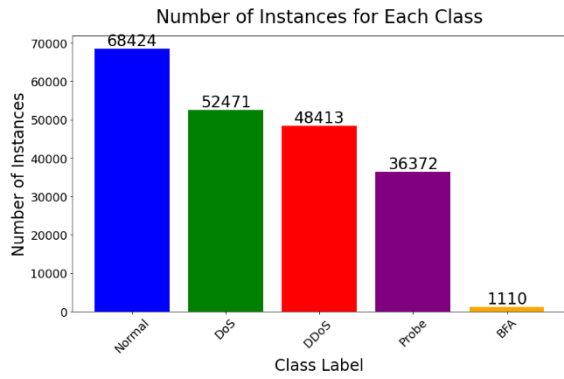


Figure 1. Within our experimental InSDN dataset

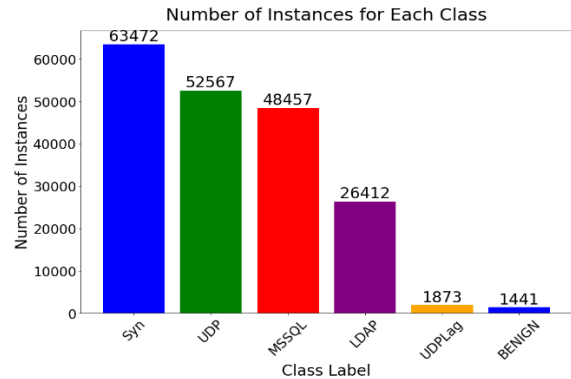


Figure 2. Within our experimental CIC-DDoS2019 dataset

### 2.2. Experimental setup

We conduct all our experiments using the Python programming language. These experiments are executed on a machine equipped with an Intel® Core™ i7-6820HQ CPU running at 2.70 GHz and 32 GB of RAM, operating on the Windows 10 operating system. Our analysis focuses on utilizing the InSDN benchmark dataset for performing multi-class traffic classification, as well as the CIC-DDoS2019 dataset for the same purpose.

### 2.3. Method for model construction and selection

This section outlines our approach for constructing our model and selecting the optimal one. Each model incorporates multiple trained k-means models, as well as trained Word2Vec and NN models. Figure 3 illustrates our method.

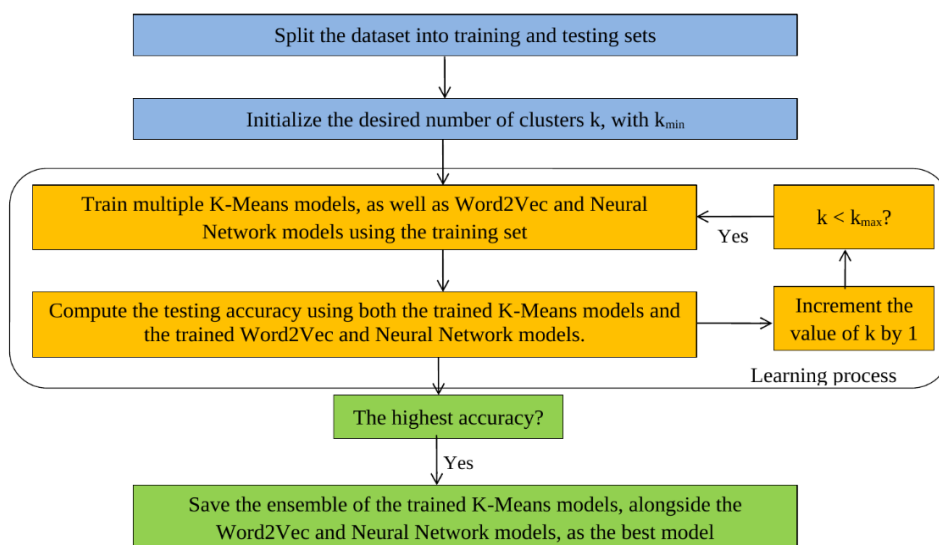


Figure 3. Flow diagram of the proposed method

For each  $k$  value within a specified range, we build a model and evaluate its accuracy using the testing set. The model with the highest testing accuracy is selected as optimal. The steps for constructing a model using our approach are as follows:

- Step 1:  $k$ -means models are trained using the training dataset.  $K$ -means [19], [20] is a fundamental clustering algorithm widely used in unsupervised learning to organize data into cohesive groups. In our method,  $k$ -means models are applied at the column level for all features except the target variable 'Label'. The number of  $k$ -means models matches the number of features our model utilizes. The Algorithm 1 partitions the data within each feature into up to  $k$  cohesive clusters. This approach treats each feature independently, allowing the algorithm to identify patterns and groups within each feature separately, without being influenced by relationships between different features. To expedite the training time of  $k$ -means models, parallel processing is employed using the parallel function from the joblib library. After this step, we assign a cluster to each data point within every column using its corresponding  $k$ -means model. Subsequently, the data in each column is converted into discrete categories labeled as 'Attribute\_ClusterX', where 'X' represents the assigned cluster ID, and 'Attribute' denotes the feature's name with spaces replaced by underscores. This transformation aims to convert numerical data into categorical data.

#### Algorithm 1. Proposed traffic classification approach

```

Input: Training set and Testing set
Output: best_K_Means_models, best_Word2Vec_model, and best_NN_model
Initialize best_accuracy = 0;
Initialize best_KMeans_models;
Initialize best_Word2Vec_model;
Initialize best_NN_model;
Define X as the matrix of feature values and y as the vector of target values;
for each k in a predefined range do
    Step 1: Train K-Means models KMeans_modelsk using Parallel Processing and Assign
Discrete
    Categories
    for each column in Xtrain do
        Train a K-Means model KMeans_modelk,column on Xtrain[column];
        Append KMeans_modelk,column to KMeans_modelsk;
    end for
    Assign clusters to Xtrain using KMeans_modelsk;
    Label each column's data with 'Attribute_ClusterX', where 'X' denotes the cluster ID
from
    KMeans_modelsk and 'Attribute' represents the feature name with spaces replaced by
underscores;
    Step 2: Create Text Feature
    Add a new column named 'text' to the dataset containing the categorical data. Each
value
    in this
    column is a list of string representations of the discrete groups;
    Step 3: Train Word2Vec Model
    Train a Word2Vec model Word2Vec_modelk on the text feature with vector size=300 and
parallel
    processing;
    Step 4: Generate Word Embeddings
    for each text entry do
        Extract word vectors for valid words in the vocabulary of Word2Vec_modelk;
        Calculate the mean of these word vectors to obtain a single vector
representation;
        Append a zero vector if no valid word vectors are found;
    end for
    Step 5: Train Neural Network Model
    Define the Neural Network architecture: two hidden layers (64 units each, ReLU
activation) and an output layer (softmax activation);
    Compile the model with Adam optimizer and sparse categorical cross-entropy loss;
    Train the Neural Network Model NN_modelk using word embeddings from Word2Vec_modelk
as input features. Use early stopping to prevent overfitting during training;
    Step 6: Evaluate on Testing Set
    Compute testing accuracy accuracyk;
    if accuracyk > best_accuracy then
        Update best_accuracy to accuracyk;
        Update best_KMeans_models to KMeans_modelsk;
        Update best_Word2Vec_model to Word2Vec_modelk;
        Update best_NN_model to NN_modelk;
    end if
end for
return best_KMeans_models, best_Word2Vec_model, and best_NN_model;

```

- Step 2: we create a text feature to serve as input for the Word2Vec model. This involves adding a new column named 'text' to the dataset obtained in the previous step, which contains the categorical data, except the last column that contains the labels. In this new 'text' column, each entry is generated by converting the categorical data (excluding the last column containing labels) from each row into strings and combining them into a list. This list represents each row of categorical data as a list of strings, which can then be processed by the Word2Vec model.
- Step 3: train a Word2Vec model using the text feature. The Word2Vec technique is widely used in natural language processing for textual analysis [21], [22]. This model learns continuous vector representations of words within the text data, capturing semantic relationships among words in a specified context. The `vector_size` parameter determines the dimensionality of the dense vectors used to represent each word. We chose a dimension of 300 based on previous research, which has indicated its effectiveness [23], [24]. During training, the model analyzes a window of five words before and after each target word to understand its context. Parallel processing with four threads is employed to expedite the training process.
- Step 4: we generate word embeddings from text data using the trained Word2Vec model. We iterate through each text in the input data, extracting word vectors for valid words found in the Word2Vec model's vocabulary. For each text, we calculate the mean of these word vectors to obtain a single vector representation of the text. If no valid word vectors are found for a text, we append a zero vector.
- Step 5: the resulting word embeddings are stored in a numpy array and utilized as input features for training the NN model. Our NN architecture consists of three dense layers: two hidden layers, each containing 64 units with ReLU activation functions, and an output layer with units equal to the number of unique classes in the target variable. The output layer uses softmax activation to generate class probabilities. We utilize the Adam optimizer and sparse categorical cross-entropy loss function for model compilation, with accuracy as the evaluation metric. The stochastic nature of the Adam optimization algorithm can result in variability in the accuracy of the NN model across iterations. Adam updates model parameters based on gradients computed from training data batches, introducing randomness into the optimization process and resulting in performance fluctuations. During training, early stopping is incorporated to prevent overfitting, with validation data used to monitor performance. This approach ensures the robustness of our NN architecture by effectively mitigating overfitting risks.

### 3. RESULTS AND DISCUSSION

This section outlines the evaluation process used to assess the effectiveness of the proposed approach. We explain the method and procedures employed during experimentation. Furthermore, we present the results from these experiments, offering valuable insights into the method's performance and capabilities.

#### 3.1. Evaluation process and criteria

The evaluation process of our approach consists of three distinct experiments, each repeated 20 to 25 times. In each iteration, we modify the desired number of clusters (represented by  $k$ ) and train our model using the specified  $k$  value. In contrast, the NN model remains unchanged across all iterations.

For all experiments, both our model and the NN model are trained using the same training dataset. The NN model shares the same architecture as ours for classification. Each model's performance is evaluated on the same testing set, using standard measures including accuracy, precision, recall, and F1-score. The first experiment, conducted on the InSDN dataset, is performed without feature selection to assess whether clustering using  $k$ -means and feature extraction using Word2Vec enhance the performance of the NN model. The second experiment, also conducted on the InSDN dataset, is similar to the first one, but we used only the relevant features (detailed in Table 1). We utilized `SelectKBest`, a feature selection module from the `scikit-learn` library, to determine the optimal number of features. The third experiment, conducted using the CIC-DDoS2019 dataset, aims to evaluate the adaptability of our method to environments beyond SDN. It is carried out without feature selection.

#### 3.2. Experimental results

This section presents the results obtained from the experiments conducted with the proposed method. Firstly, by varying the value of  $k$ , we compare the accuracy achieved by the NN model and our proposed model. We then compare the best proposed model with the best NN model, both evaluated at their highest accuracy, using precision, recall, and F1-score. Finally, we analyze the training time for each model trained with our approach on the InSDN dataset. Figures 4 and 5 depict the performance on our InSDN dataset without feature selection. Figure 4 illustrates accuracy across iterations with varying  $k$  values, while Figure 5 compares classification reports. Figures 6 and 7 show the impact of feature selection on our InSDN dataset, with Figure 6 displaying accuracy and Figure 7 providing comparison details. Finally,

Figures 8 and 9 present the performance on our CIC-DDoS2019 dataset without feature selection. Figure 8 showcases accuracy trends, and Figure 9 compares classification reports.

Table 1. Extracted subset features from our experimental InSDN dataset

No.	Feature	No.	Feature	No.	Feature
1	Protocol	15	Fwd IAT Max	29	SYN Flag Cnt
2	Flow Duration	16	Bwd IAT Tot	30	PSH Flag Cnt
3	Fwd Pkt Len Max	17	Bwd IAT Mean	31	ACK Flag Cnt
4	Fwd Pkt Len Min	18	Bwd IAT Std	32	URG Flag Cnt
5	Bwd Pkt Len Max	19	Bwd IAT Max	33	Down/Up Ratio
6	Bwd Pkt Len Min		Bwd PSH Flags	34	Pkt Size Avg
7	Bwd Pkt Len Mean		Bwd URG Flags	35	Fwd Seg Size Avg
8	Bwd Pkt Len Std		Fwd Pkts/s	36	Bwd Seg Size Avg
9	Flow Pkts/s		Bwd Pkts/s	37	Init Bwd Win Byts
10	Flow IAT Std		Pkt Len Min	38	Idle Mean
11	Flow IAT Max		Pkt Len Max	39	Idle Std
12	Fwd IAT Tot		Pkt Len Mean	40	Idle Max
13	Fwd IAT Mean		Pkt Len Std	41	Idle Min
14	Fwd IAT Std		FIN Flag Cnt		

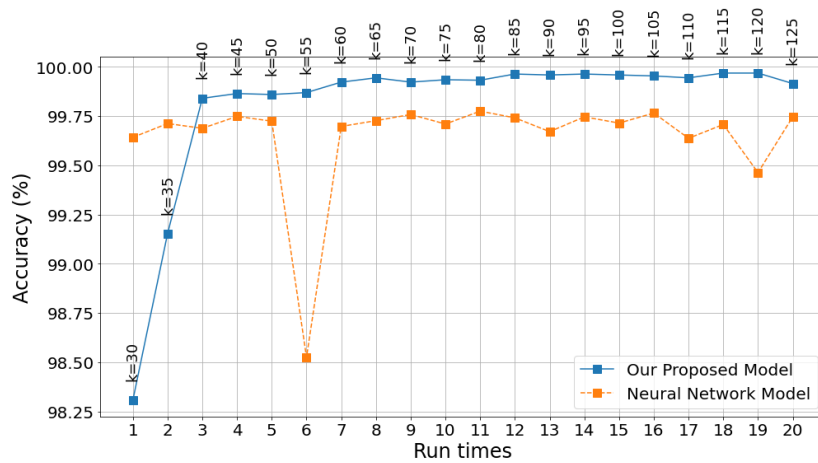


Figure 4. Accuracy in our InSDN dataset (no feature selection)

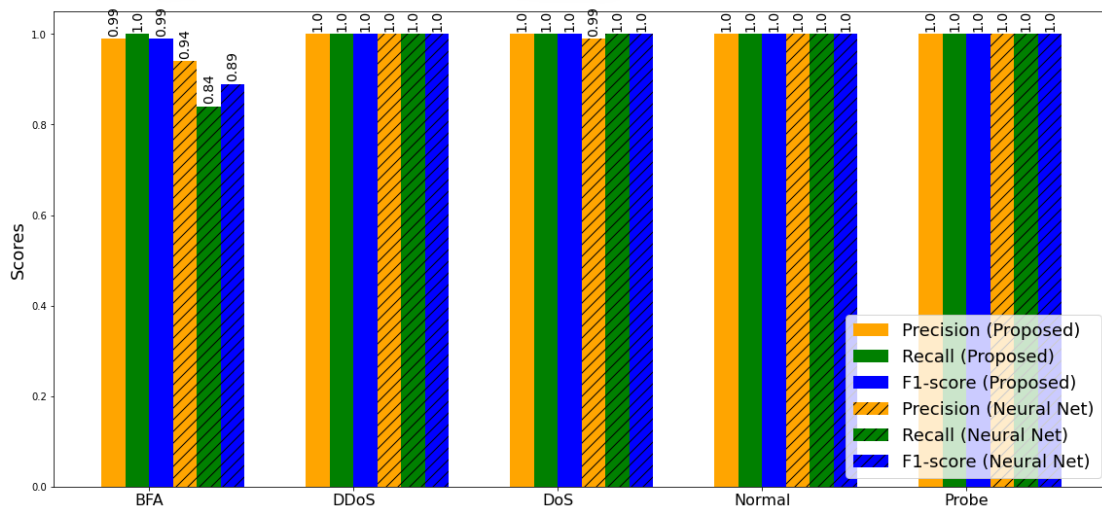


Figure 5. Classification reports comparison in our InSDN dataset (no feature selection)

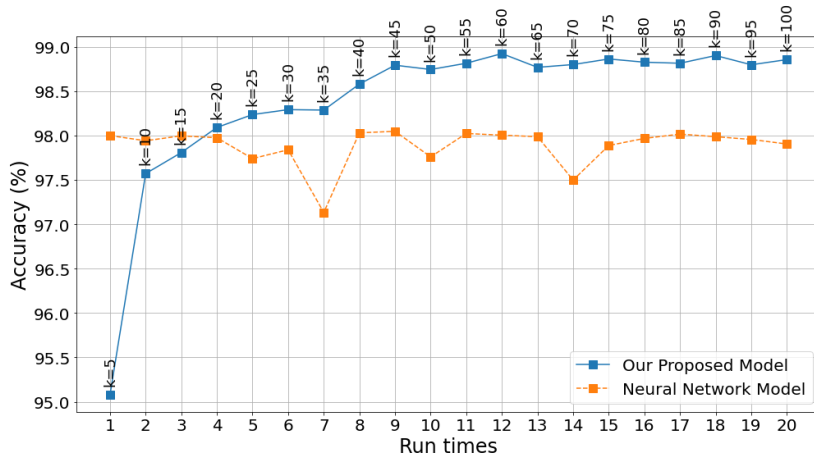


Figure 6. Accuracy in our InSDN dataset (with feature selection)

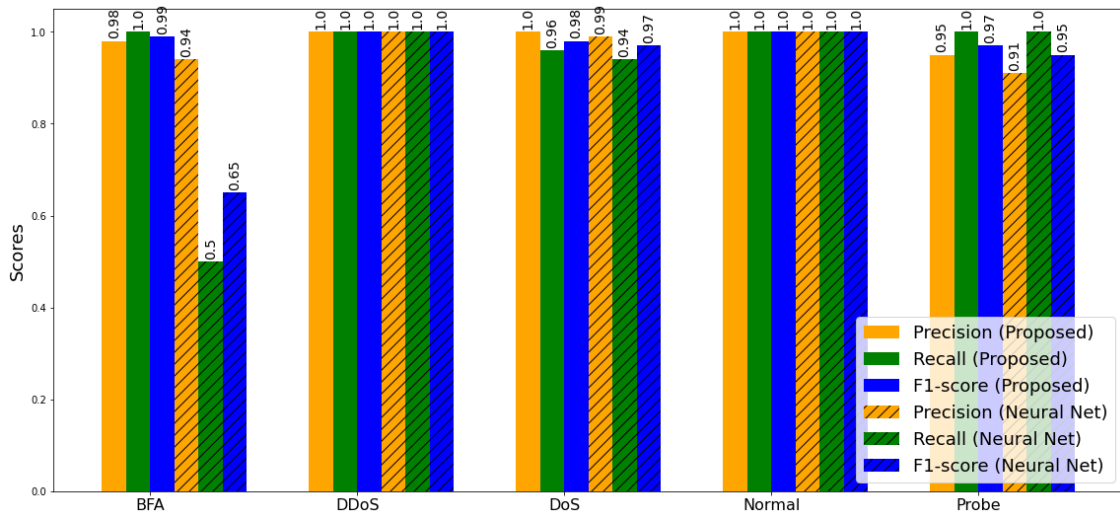


Figure 7. Classification reports comparison in our InSDN dataset (with feature selection)

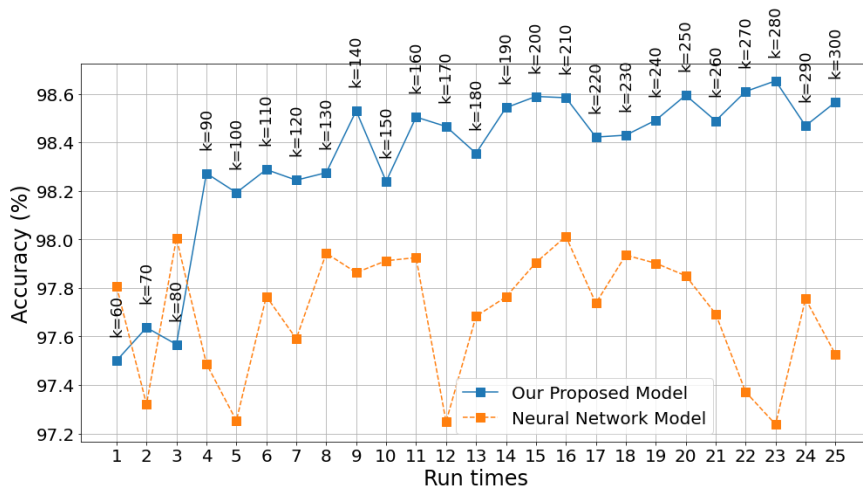


Figure 8. Accuracy in our CIC-DDoS2019 dataset (no feature selection)

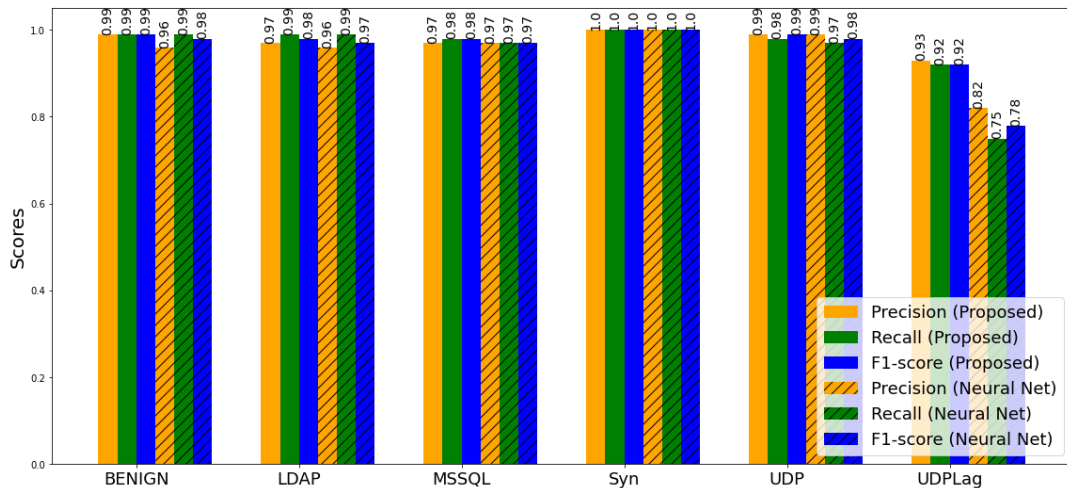


Figure 9. Classification reports comparison in our CIC-DDoS2019 dataset (no feature selection)

**3.2.1. Evaluation on our experimental InSDN dataset without feature selection**

As depicted in Figure 4, once surpassing the approximate threshold of k=40, two key observations become evident: firstly, our proposed model demonstrates superior accuracy compared to the NN model. Furthermore, its accuracy continues to improve, notably, it reaches its peak around k=115, achieving exceptional accuracy. Additionally, starting from around k=65 onwards, the accuracy appears to stabilize, indicating consistent performance of our proposed model even with varying values of k. For this reason, we'll next focus our model comparison from run time 8, corresponding to k=65.

According to Table 2, our model achieves a higher maximum accuracy of 99.97% compared to the NN model's maximum accuracy of 99.77%. Additionally, our model exhibits a higher mean accuracy of 99.95% compared to the NN model's mean accuracy of 99.70%. Moreover, our model demonstrates a smaller variance of 0.0003, indicating more consistent performance, while the NN model has a variance of 0.0067, suggesting greater variability in accuracy. This pronounced superiority underscores the effectiveness and robustness of our proposed approach, which offers superior accuracy and stability across a wide range of desired number of clusters compared to the NN model.

Table 2. Accuracy of our model and NN model over iterations (%) for our InSDN without feature selection

Iteration number	Our model	NN model	Iteration number	Our model	NN model
1	98.3074	99.6421	11	99.9322	99.7751
2	99.1537	99.7122	12	99.9637	99.7412
3	99.8404	99.6880	13	99.9588	99.6711
4	99.8645	99.7485	14	99.9637	99.7461
5	99.8597	99.7243	15	99.9588	99.7146
6	99.8694	98.5250	16	99.9540	99.7654
7	99.9226	99.6977	17	99.9443	99.6373
8	99.9443	99.7267	18	99.9685	99.7074
9	99.9226	99.7582	19	99.9685	99.4632
10	99.9347	99.7098	20	99.9153	99.7461

Analysis of the classification report for our model with k=115, as depicted in Figure 5, reveals superior performance in precision, recall, and F1-score across all classes. Notably, our model achieves a precision of 1 for classifying DoS attacks, ensuring accurate identification. Similarly, for the Normal traffic class, our model demonstrates precision, recall, and F1-score values of 1, underscoring its proficiency in detecting normal network behavior. On the other hand, the NN model demonstrates comparable performance, with a marginal difference in precision for the DoS class and lower precision and recall values for the BFA class, decreasing to 0.94 and 0.84 respectively. This suggests challenges in accurately identifying instances of BFA attacks. These comparisons underscore the robustness and effectiveness of our proposed model in accurately classifying network traffic, encompassing various attack types as well as normal traffic.



### 3.2.2. Evaluation on our experimental InSDN dataset with feature selection

Figure 6 illustrates the superior accuracy of our model compared to the NN model, evident from approximately  $k=20$  onwards. Beyond  $k=45$ , the accuracy exhibits marginal deviations without significant improvement. Therefore, we focus our analysis starting from run time 9, corresponding to  $k=45$ .

From Table 3, our model achieves a maximum accuracy of 98,92%, with a mean accuracy of approximately 98,82% and a variance of approximately 0,0027. In the other hand, the NN model reaches a maximum accuracy of 98,05%, with a mean accuracy of approximately 97,92% and a variance of approximately 0,0237. These results highlight that our model not only achieves a higher maximum accuracy but also demonstrates a more consistent and stable performance compared to the NN model, as evidenced by its notably lower variance.

Table 3. Accuracy of our model and NN model over iterations (%) for our InSDN with feature selection

Iteration number	Our model	NN model	Iteration number	Our model	NN model
1	95.0795	97.9979	11	98.8128	98.0221
2	97.5699	97.9399	12	98.9216	98.0028
3	97.8069	97.9955	13	98.7668	97.9858
4	98.0898	97.9737	14	98.7982	97.4974
5	98.2349	97.7392	15	98.8611	97.8867
6	98.2905	97.8383	16	98.8248	97.9665
7	98.2857	97.1323	17	98.8128	98.0148
8	98.5782	98.0294	18	98.8998	97.9858
9	98.7910	98.0463	19	98.7958	97.9544
10	98.7426	97.7585	20	98.8539	97.9036

As depicted in Figure 7, our model with  $k=60$  demonstrates superior precision across all classes compared to the NN model. With precision scores ranging from 0.95 to 1.00, our model showcases its ability to accurately classify instances, even for the BFA class, which contains a small number of instances. Additionally, our model exhibits consistently higher recall rates, ranging from 0.96 to 1.00 across different classes. This difference is particularly evident in the BFA class, where it achieves a recall of 1, compared to the NN model's 0.5. Furthermore, the balanced performance across precision and recall is reflected in the F1-scores, consistently favoring our model over the NN model, with F1-scores ranging from 0.97 to 1.00 across all classes. These findings underscore the robustness of our proposed model in accurately classifying network traffic types, crucial for effective cybersecurity measures in real-world scenarios.

### 3.2.3. Evaluation on our experimental CIC-DDoS2019 dataset

The results depicted in Figure 8 highlight the promising performance of our proposed model on the CIC-DDoS2019 dataset. Across the range of experiments conducted with varying desired numbers of clusters, our model began to outperform the NN model in accuracy from around  $k=90$  onwards. The accuracy demonstrated greater stability and consistently remained above 98.4% from approximately  $k=190$ , corresponding to run time 14. Consequently, we'll conduct our comparison only from iteration number 14.

Based on Table 4, our model exhibits a variance of 0.0056, a mean accuracy of 98.54%, and a maximum accuracy of 98.65%. In contrast, the NN model shows a higher variance of 0.0556, a mean accuracy of 97.72%, and a maximum accuracy of 98.01%. These results confirm our model's superior accuracy compared to the NN model and underscore our model's enhanced stability.

Table 4. Accuracy of our model and NN model over iterations (%) for our CIC-DDoS2019 without feature selection

Iteration number	Our model	NN model	Iteration number	Our model	NN model
1	97.5003	97.8066	14	98.5429	97.7629
2	97.6367	97.3201	15	98.5892	97.9044
3	97.5672	98.0048	16	98.5841	98.0126
4	98.2726	97.4874	17	98.4219	97.7371
5	98.1928	97.2531	18	98.4296	97.9353
6	98.2880	97.7654	19	98.4914	97.9019
7	98.2443	97.5904	20	98.5944	97.8504
8	98.2751	97.9431	21	98.4888	97.6933
9	98.5300	97.8633	22	98.6098	97.3716
10	98.2391	97.9122	23	98.6536	97.2377
11	98.5043	97.9250	24	98.4682	97.7551
12	98.4656	97.2480	25	98.5660	97.5260
13	98.3550	97.6830			

Figure 9 illustrates a comparison between the classification reports of our model with  $k=280$  and the NN model, revealing significant differences in their performance metrics. Notably, our model demonstrates superior precision, recall, and F1-score across multiple classes, indicating its effectiveness in accurately classifying network traffic. Classes such as BENIGN, LDAP, MSSQL, Syn, and UDP consistently exhibit higher performance metrics in our model compared to the NN model. Even within the UDPLag class, where instances are fewer in number, our model consistently exhibits superior precision, recall, and F1-score compared to the NN model. Overall, our model demonstrates more robust performance across all classes, confirming its effectiveness in network traffic classification.

### 3.2.4. Computational overhead on our experimental InSDN dataset

The multi-step process of our proposed approach might introduce significant computational overhead. This is because, for each  $k$  value, we construct a model that combines multiple  $k$ -means models, Word2Vec model, and a NN classifier to identify the one with the highest accuracy. As depicted in Figures 10 and 11, the training times show considerable variability and generally increase with the desired number of clusters. Without feature selection, the proposed model exhibits significantly higher training times, ranging from approximately 328.79 seconds to 841.08 seconds. In contrast, when feature selection is applied, the proposed model experiences reduced training times, decreasing to a range of 158.40 seconds to 449.53 seconds. Overall, feature selection has a substantial positive impact, effectively reducing training times and enhancing the efficiency of the proposed method.



Figure 10. Training time comparison without feature selection



Figure 11. Training time comparison with feature selection

### 3.3. Analysis and discussion

The experimental results show that our approach excels in both accuracy and stability compared to traditional NN models across various datasets. On the InSDN dataset, our method achieves a superior accuracy of 99.97% without feature selection when the desired number of clusters exceeds approximately 40, outperforming robust approaches such as the deep learning method in paper [25], the GRU model in paper [26], and the stacked auto-encoder multi-layer perceptron (SAE-MLP) classifier used in paper [27], which achieved accuracy rates of 96%, 99.65%, and 99.75%, respectively. It also maintains higher accuracy with

feature selection, especially after  $k \approx 45$ . Similarly, on the CIC-DDoS2019 dataset, our approach demonstrates strong accuracy, with a stable performance trend from approximately  $k=190$  onwards, averaging around 98.54%. This stability, along with high precision, recall, and F1-scores across various traffic types, highlights the method's effectiveness in handling complex network conditions, even with limited sample sizes.

The multi-step process of training multiple k-means models, along with training Word2Vec and NN models, introduces significant computational overhead that increases with the desired number of clusters. However, feature selection reduces some of this overhead and enhances efficiency. Future improvements should focus on addressing this computational overhead by utilizing advanced parallel processing, cloud-based computing resources, and optimizing system architecture. These enhancements will improve the model's scalability and efficiency, making it more suitable for real-world applications.

The techniques developed in this study have potential benefits for other fields requiring high classification accuracy, particularly where dealing with imbalanced datasets is challenging. Such techniques could be valuable in areas like medical diagnostics or fraud detection, where precise classification is crucial despite uneven data distributions.

#### 4. CONCLUSION

This study underscores the critical importance of feature-based approaches in enhancing SDN security through advanced clustering algorithms and feature extraction techniques. Starting from a specific desired number of clusters, our model, which integrates multiple k-means models for clustering, Word2Vec for feature extraction, and NN for traffic classification, consistently achieves superior accuracy and stability across diverse traffic classes, including those with limited sample sizes. These findings highlight the potential of our approach to effectively capture and categorize intricate patterns within SDN environments, surpassing the performance of traditional NN model. However, the computational demands associated with employing multiple k-means models and Word2Vec emphasize the need for efficient parallelization and robust infrastructure to ensure scalability and practical deployment in real-world scenarios. Looking ahead, our future research will focus on optimizing our approach to enhance the model's effectiveness and applicability. We plan to evaluate the model's performance in real-world networks, providing insight into its operational capabilities. Additionally, we will explore alternative feature extraction methods beyond Word2Vec, investigate different clustering algorithms to optimize performance further, and experiment with additional machine learning techniques for classification. By refining these aspects, we aim to make our approach more versatile and capable of addressing the evolving security needs of SDN environments.

#### ACKNOWLEDGEMENTS

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors.




#### REFERENCES

- [1] E. Haleplidis, E. K. Pentikousis, S. Denazis, J. H. Salim, D. Meyer, and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology," *Internet Research Task Force (IRTF)*, pp. 1–35, Jan. 2015.
- [2] T. Mizrahi, N. Sprecher, E. Bellagamba, and Y. Weingarten, "An Overview of Operations, Administration, and Maintenance (OAM) Tools," RFC 7276, 2014, doi: 10.17487/rfc7276.
- [3] L. Zheng, J. Jiang, W. Pan, and H. Liu, "High-performance and range-supported packet classification algorithm for network security systems in SDN," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, Dublin, Ireland, 2020, pp. 1–6, doi: 10.1109/ICCWorkshops49005.2020.9145461.
- [4] L. Tan, Y. Pan, J. Wu, J. Zhou, H. Jiang, and Y. Deng, "A New Framework for DDoS Attack Detection and Defense in SDN Environment," *IEEE Access*, vol. 8, pp. 161908–161919, 2020, doi: 10.1109/ACCESS.2020.3021435.
- [5] Y. Xu, H. Sun, F. Xiang, and Z. Sun, "Efficient DDoS Detection Based on K-FKNN in Software Defined Networks," *IEEE Access*, vol. 7, pp. 160536–160545, 2019, doi: 10.1109/ACCESS.2019.2950945.
- [6] T. Jafarian, M. Masdari, A. Ghaffari, and K. Majidzadeh, "SADM-SDNC: security anomaly detection and mitigation in software-defined networking using C-support vector classification," *Computing*, vol. 103, no. 4, pp. 641–673, Apr. 2021, doi: 10.1007/s00607-020-00866-x.
- [7] S. Garg, K. Kaur, N. Kumar, and J. J. P. C. Rodrigues, "Hybrid deep-learning-based anomaly detection scheme for suspicious flow detection in SDN: A social multimedia perspective," *IEEE Transactions on Multimedia*, vol. 21, no. 3, pp. 566–578, Mar. 2019, doi: 10.1109/TMM.2019.2893549.
- [8] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, M. Ghogho, and F. El Moussa, "DeepIDS: Deep learning approach for intrusion detection in software defined networking," *Electronics*, vol. 9, no. 9, pp. 1–18, Sep. 2020, doi: 10.3390/electronics9091533.
- [9] M. Tavallaei, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the KDD CUP 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, ON, Canada, 2009, pp. 1–6, doi: 10.1109/CISDA.2009.5356528.
- [10] N. S. Shaji, T. Jain, R. Muthalagu, and P. M. Pawar, "Deep-discovery: Anomaly discovery in software-defined networks using




- artificial neural networks,” *Computers and Security*, vol. 132, p. 103320, Sep. 2023, doi: 10.1016/j.cose.2023.103320.
- [11] J. V. Staden and D. Brown, “An Evaluation of Machine Learning Methods for Classifying Bot Traffic in Software Defined Networks,” in *Proceedings of Third International Conference on Sustainable Expert Systems: ICSES 2022*, Singapore: Springer Nature Singapore, 2023, pp. 979–991, doi: 10.1007/978-981-19-7874-6\_72.
- [12] M. N. Jasim and M. T. Gaata, “K-Means clustering-based semi-supervised for DDoS attacks classification,” *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 6, pp. 3570–3576, Dec. 2022, doi: 10.11591/eei.v11i6.4353.
- [13] J. Cui, J. Zhang, J. He, H. Zhong, and Y. Lu, “DDoS detection and defense mechanism for SDN controllers with K-Means,” in *2020 IEEE/ACM 13th International Conference on Utility and Cloud Computing (UCC)*, Leicester, UK, 2020, pp. 394–401, doi: 10.1109/UCC48980.2020.00062.
- [14] M. S. Elsayed, N. A. Le-Khac, and A. D. Jurcut, “InSDN: A novel SDN intrusion dataset,” *IEEE Access*, vol. 8, pp. 165263–165284, 2020, doi: 10.1109/ACCESS.2020.3022633.
- [15] “M. S. Elsayed, N.-A. Le-Khac, A. Jurcut, 2020, ‘InSDN: SDN Intrusion Dataset,’ UCD School of Computer Science, ASEADOS Lab. Available: <http://aseados.ucd.ie/datasets/SDN/>.
- [16] I. Sharafaldin, A. H. Lashkari, S. Hakak, and A. A. Ghorbani, “DDoS evaluation dataset (CIC-DDoS2019),” 2019. [Online]. Available: <https://www.unb.ca/cic/datasets/ddos-2019.html>.
- [17] H. Yzzogh, “InSDN\_Traffic\_Analysis.rar,” GitHub, Available: <https://github.com/Yzzogh/Traffic-Analysis>. (Accessed: Feb. 03, 2024).
- [18] H. Yzzogh, “CIC-DDoS2019\_Traffic\_Analysis.rar,” Github, Available: [https://github.com/Yzzogh/Traffic-Analysis/blob/main/CIC-DDoS2019\\_Traffic\\_Analysis.rar](https://github.com/Yzzogh/Traffic-Analysis/blob/main/CIC-DDoS2019_Traffic_Analysis.rar), (Accessed: Feb. 04, 2024).
- [19] M. Ahmed, R. Seraj, and S. M. S. Islam, “The k-means algorithm: A comprehensive survey and performance evaluation,” *Electronics*, vol. 9, no. 8, pp. 1–12, Aug. 2020, doi: 10.3390/electronics9081295.
- [20] T. M. Ghazal *et al.*, “Performances of k-means clustering algorithm with different distance metrics,” *Intelligent Automation and Soft Computing*, vol. 30, no. 2, pp. 735–742, 2021, doi: 10.32604/iasc.2021.019067.
- [21] K. W. Church, “Word2Vec,” *Natural Language Engineering*, vol. 23, no. 1, pp. 155–162, Jan. 2017, doi: 10.1017/S1351324916000334.
- [22] D. E. Cahyani and I. Patasik, “Performance comparison of tf-idf and word2vec models for emotion text classification,” *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 5, pp. 2780–2788, Oct. 2021, doi: 10.11591/eei.v10i5.3157.
- [23] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv*, 2013, doi: 10.48550/arXiv.1301.3781.
- [24] G. Rao, W. Huang, Z. Feng, and Q. Cong, “LSTM with sentence representations for document-level sentiment classification,” *Neurocomputing*, vol. 308, pp. 49–57, Sep. 2018, doi: 10.1016/j.neucom.2018.04.045.
- [25] A. Malik, R. De Frein, M. Al-Zeyadi, and J. Andreu-Perez, “Intelligent SDN Traffic Classification Using Deep Learning: Deep-SDN,” in *2020 2nd International Conference on Computer Communication and the Internet (ICCCI), Nagoya, Japan*, 2020, pp. 184–189, doi: 10.1109/ICCCI49374.2020.9145971.
- [26] D. Nuñez-Agurto, W. Fuertes, L. Marrone, E. Benavides-Astudillo, C. Coronel-Guerrero, and F. Perez, “A Novel Traffic Classification Approach by Employing Deep Learning on Software-Defined Networking,” *Future Internet*, vol. 16, no. 5, pp. 1–23, Apr. 2024, doi: 10.3390/fi16050153.
- [27] N. Ahuja, G. Singal, and D. Mukhopadhyay, “DLSDN: Deep learning for DDOS attack detection in software defined networking,” in *2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, 2021, pp. 683–688, doi: 10.1109/Confluence51648.2021.9376879.

## BIOGRAPHIES OF AUTHORS



**Hicham Yzzogh**    received the engineering degree in telecommunications from INPT, Morocco, in 2005. With over 18 years of experience at Nokia, he has worked across diverse core network environments and possesses strong skills in coding and machine learning algorithms. He is certified as an Azure Solutions Architect Expert and is currently pursuing a Ph.D. at Mohammed V University in Rabat, Morocco. His research interests include network security, SDN, natural language processing, and image processing. He can be contacted at email: [hicham\\_yzzogh@um5.ac.ma](mailto:hicham_yzzogh@um5.ac.ma).



**Hafssa Benaboud**    received her Ph.D. degree in Computer Sciences from Burgundy University Dijon-France in 2004. In 2005, she joined as an Assistant Professor at Applied Sciences National School (ENSA) of Tangier, Morocco, and has been working as a Full Professor since 2011 in the Department of Computer Sciences at Mohammed V University in Rabat, Morocco. She has authored more than 30 articles published in international journals and international conference proceedings. Her research interests include network protocols, network security, internet of things, traffic analysis and quality of services. She can be contacted at email: [hafssa.benaboud@fsr.um5.ac.ma](mailto:hafssa.benaboud@fsr.um5.ac.ma).