Arabian Journal for Science and Engineering (2022) 47:9965–9983
https://doi.org/10.1007/s13369-021-06484-9

RESEARCH ARTICLE-COMPUTER ENGINEERING AND COMPUTER SCIENCE

VMFCVD: An Optimized Framework to Combat Volumetric DDoS Attacks using Machine Learning

Arvind Prasad[1] · **Shalini Chandra**[1]

Received: 12 August 2021 / Accepted: 6 December 2021 / Published online: 23 January 2022

Abstract
Despite significant development in distributed denial of service (DDoS) defense systems, the downtime caused by DDoS damages reputation, crushes end-user experience, and leads to considerable revenue loss. Volumetric DDoS attacks are the most common form of DDoS attack and are carried out by an army of infected IoT devices or by reflector servers, which increase attacks at massive scales. In this work, we propose a voting-based multimode framework to combat volumetric DDoS (VMFCVD) attacks. VMFCVD is based on a triad of fast detection mode (FDM), defensive fast detection mode (DFDM), and high accuracy mode (HAM) methods. FDM is designed to classify network traffic when the server is under attack. The highly dimensionally reduced dataset helps FDM accelerate detection speed. During our experiment, the dimension reduction for FDM was more than 97% while maintaining an accuracy of 99.9% in most cases. DFDM is an extended version of FDM that enhances malicious network traffic detection accuracy by tightening the detection technique. HAM focuses on detection accuracy, showing substantial improvement over FDM and DFDM. HAM activates when the server is stable. VMFCVD is extensively experimented on the latest benchmark DDoS and botnet datasets, namely the CICIDS2017 (BoT & DDoS), CSE-CIC-IDS2018 (BoT & DDoS), CICDDoS2019 (DNS, LDAP, SSDP & SYN), DoHBrw2020, NBaIoT2018 (Mirai), UNSW2018 BoTIoT, and UNSW NB15 datasets. The VMFCVD results show that it outperforms recent studies. VMFCVD performs exceptionally well when the server is under DDoS attack.

Keywords Cyber security · Volumetric DDoS · Botnet attack · Internet of Things · Machine learning · Feature engineering

1 Introduction

The COVID-19 outbreak and rapid spread of the coronavirus have driven organizations to shift their services online. In the wake of the epidemic, sectors, such as education, health care, entertainment, food services, and retail, have moved online to contain the spread of the virus. Most organizations have implemented work from home either fully or partially for the same reason. The digital shift underwent a quantum leap within a span of a few months. With the growth of dependencies on online services and activities due to emergency scenarios, it has become crucial for organizations to provide uninterrupted services/resources to end users. Unavailability of services/resources can incur a massive loss for organizations. DDoS attacks are one of the most effective attacks used by cybercriminals to prevent legitimate users from using the service by exhausting server resources. DDoS attacks have become more sophisticated with tremendous increases in volume [1]. In recent years, Internet of Things (IoT)-enabled DDoS attacks have increased at an alarming level. Cybercriminals are launching DDoS attacks using botnets (robot networks), an army of infected IoT devices, which increase the intensity of attacks to tbps. Botnets work like an army to launch a DDoS attack on a victim server through a centralized botnet command and control system [2]. The modern DDoS attack has become a mainstream commodity in the cyber world [3]. There are underground communities of cybercriminals [2], where each cybercriminal has an army of bots. They team up to launch DDoS attacks with much higher volume and velocity on the victim server.

DDoS attacks can be direct attacks or an amplified reflection-based attack on the victim server [4]. Direct attacks are carried out by an army of infected devices, while reflective servers are used for amplified reflection-based attacks.

✉ Arvind Prasad
arvindbitm@gmail.com

[1] Department of Computer Science, Babasaheb Bhimrao Ambedkar University (A Central University), Lucknow 226025, UP, India

Researcher [5] conducted a systematic mapping study to evaluate the most common cyber security threats by analyzing 78 primary studies. They reported denial of service as the most addressed vulnerability with the frequency of 37% in their systematic mapping study. F5Lab reported that 73% of DDoS attacks are volumetric DDoS attacks, out of which 53% are reflection-based attacks launched using vulnerable servers [6] or infected devices. Volumetric DDoS attacks exploit network protocol fragility to target servers. It overwhelms the server's resources by sending massive traffic or service requests. Network protocol servers such as SSDP (simple service discovery protocol) and LDAP (lightweight directory access protocol) servers are well-known examples of volumetric DDoS attacks. The amplified reflection-based SSDP attack generates an enormous amount of traffic by exploiting the UPnP (universal plug and play) protocol. LDAP servers supporting UDP (user datagram protocol) services are used to launch volumetric LDAP DDoS attacks.

SYN flood attacks are another example of a volume-based DDoS attack. Attackers abuse the 3-way handshaking process of stateful TCP (transmission control protocol) connections to launch SYN flood attacks by repeatedly sending many SYN packets to the victim servers. Servers must reply to each SYN packet with the SYN-ACK (synchronize–acknowledge) flag and wait for each SYN request's acknowledgment packet. This process uses some memory and processing power of a server. The flood of SYN requests overwhelm servers, resulting in service unavailable errors to any new request, including legitimate users. The connectionless nature of the UDP protocol causes the UDP protocol-based amplified reflection DDoS (AR-DDoS) attacks. AR-DDoS requires minimal effort [7] to launch an efficient volumetric DDoS attack. The connectionless nature is for the enormous benefit of the network, but attackers misuse it. They abuse UDP services such as LDAP, Memcached, NTP, and DNS to execute attacks.

With the ever-growing dependencies on the digital world, it has become vital to provide uninterrupted services to users. Unavailability of service for a fraction of time can cause considerable revenue loss for any business. Most organizations do not reveal if they are attacked by DDoS, making it difficult to estimate the financial loss caused by DDoS attacks. It has been reported that there is a 55% increase in DDoS attacks from January 2020 to March 2021 [6]. In 54% of incidents, attackers launched DDoS attacks using multiple modern attack vectors. Figure 1 depicts the modern attack vector. Despite many studies carried out by researchers [8–19] and leading organizations in cybersecurity sectors, DDoS attacks are rapidly growing and pose a tremendous threat to cyberspace.

The significant contributions of this work are summarized as follows:

- This paper proposes a weighted voting-based multimode machine learning framework, VMFCVD, to detect and mitigate volumetric DDoS attacks.
- VMFCVD has three modes, namely FDM, DFDM & high accuracy mode (HAM), to classify network packets. Initially, HAM is activated. Based on the DDoS attack possibilities, the mode switches.
- FDM has low computational and memory overhead, as it takes only two features to classify any network packet. It activates when the framework observes a high volume of incoming traffic.
- DFDM has the same low computational and memory overhead as FDM. Only if all votes are in favor of a packet, does DFDM allow it.
- HAM has more computational and memory overhead than FDM and DFDM, but gives the highest accuracy.
- We have compared the performance of VMFCVD with traditional ML algorithms and with state-of-the-art baselines.

2 Related Work

DDoS attacks have a high potential to bring down unprotected servers within a small fraction of time, making them a growing concern for all organizations committed to providing uninterrupted services to their users. Various researchers have proposed techniques to defend against DDoS attacks. In this section, we discuss ML-based classification techniques to combat DDoS attacks.

Aamir and Zaid [8] proposed an ML framework to detect DDoS attacks where they initially applied feature engineering, such as backward elimination, Chi-square test, and information gain. Feature engineering supports the dataset to overcome issues related to missing values, skewness, and collinearity–multicollinearity. They applied five ML models, namely K-nearest neighbors (KNN), naive Bayes (NB), support vector machines (SVM), random forests (RF), and artificial neural networks (ANN), to evaluate their framework's performance. To obtain optimal results, their experimental setup was different for each dataset. Doriguzzi-Corin et al. [9] proposed LUCID, a lightweight CNN-based DDoS attack detection technique intending to speed up network traffic classification on resource-containing devices. Their proposed approach produces consistent detection results on datasets such as ISCX2012, CIC2017, UNB201X, and CSE-CIC2018 with accuracies of 0.988, 0.9967, and 0.9946, respectively. They created a tool to extract network traffic into the required input format for LUCID for live detection. Jia et al. [10] presented two ML models, long short-term memory (LSTM) and convolutional neural network (CNN), to identify and classify malicious traffic. They called it Flowguard. Flowguard was validated on the CICDDoS2019

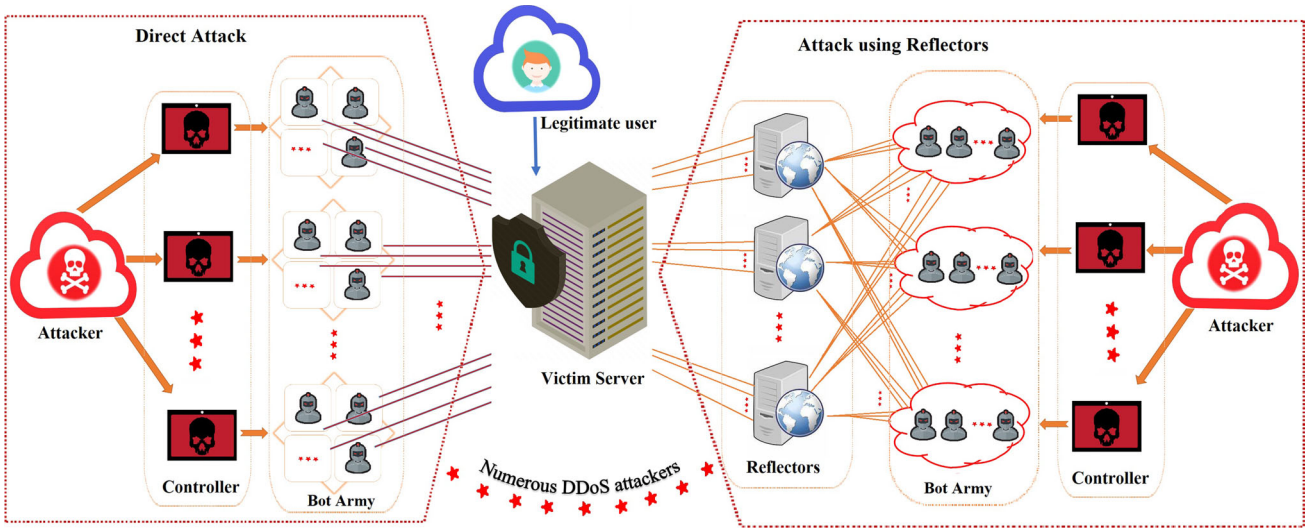

Fig. 1 A modern cyberattack overview

dataset and on their dataset generated using BoNeSi and SlowHTTPTest DDoS simulators. Their proposed model recorded an accuracy of 98.9% and outperformed other ML models implemented in work, namely ID3, random forest (RF), naive Bayes (NB), and logistic regression (LR) models.

Injadat et al. [11] proposed a multistage ML framework for network intrusion detection. The stages involved in their framework are data preprocessing, feature selection, hyperparameter optimization, and combination to give an optimized result. The main techniques used in various stages were Z score normalization & synthetic minority oversampling techniques for the first stage, mutual information gain and feature correlation for the second stage, and random search, meta-heuristic optimization algorithms, and Bayesian optimization techniques for parameter optimization in the third stage. Researchers evaluated the framework on the CICIDS 2017 and UNSW-NB 2015 datasets. They were able to enhance the detection accuracy by over 99%. Priyadarshini and Barik [12] proposed a deep learning-based DDoS attack mitigation technique to protect fog and cloud computing environments from DDoS attacks. They deployed the proposed mechanism on the SDN controller in a software-defined network. The proposed model was evaluated on the Hogzilla dataset. Researchers also evaluated it on live DDoS network traffic extracted using TCPDump. They obtained a maximum accuracy of 99.12% on the training sample and 98.88% on the test sample.

Aamir and Zaid [13] developed a clustering-based semisupervised ML scheme to improve DDoS detection. They applied agglomerative clustering and principal component analysis (PCA) with K-means clustering to reduce the dimensionality of the dataset. In the next step, they developed a voting technique to classify the label of the network traffic. They evaluated the proposed framework on the CICIDS2017 DDoS dataset, a subset of the CICIDS2017 dataset. The KNN, SVM, and RF algorithms determined the voting to classify the network traffic into benign or DDoS attacks. The KNN, SVM, and RF accuracies were 95%, 92%, and 96.66%, respectively, and the voting model obtained 82.10% accuracy. They intended to include more ML models to enhance the accuracy. Rehman et al. [14] proposed DIDDOS, a DL- and ML-based framework to detect DDoS attacks. DIDDOS was based on a gated recurrent unit (GRU), recurrent neural network (RNN) from deep learning, naive Bayes (NB), and sequential minimal optimization (SMO) from the machine learning paradigm. The experiment was performed on the CICDDoS2019 dataset. They recorded the highest accuracy of 99.91% using the GRU classifier on the CICDDoS2019 SSDP dataset. CICDDoS2019 SSDP is the subset of the CICDDoS2019 dataset with a massive number of records, where only 0.0003% of records are benign records and the remaining 99.9997% of records are SSDP DDoS attack records (763 benign and 2610611 SSDP DDoS records). Popoola et al. [15] developed a deep learning-based botnet attack detection framework called LAE-BLSTM by implementing a long short-term memory autoencoder (LAE) for dimensionality reduction and bidirectional long short-term memory (BLSTM) for the classification of network traffic into benign and malicious traffic. LAE-BLSTM experimented on the Bot-IoT dataset. LAE reduced the number of features from 37 to 6; deep BLSTM outperformed on a reduced dataset when the Nadam optimizer was applied. The framework is preferable for memory-constrained IoT devices.

Ravi and Shalinie [16] developed a learning-driven detection mitigation (LEDEM) mechanism to detect and mitigate DDoS attacks triggered by malicious IoT devices on IoT

servers. Supervised machine learning techniques were used for attack detection. They proposed different fixed IoT (fIot) and mobile IoT (mIoT) techniques for attack mitigation. An approximation algorithm, fMS, was used for fIoT, where malicious IoT devices were grouped according to their VLAN id and the packets were dropped if they belonged to that VLAN or even disconnected that VLAN. A greedy drop rule was used for mIoT, which dropped all the incoming packets from the malicious IoT devices. A testbed was created to verify LEDEM in a real network. LEDEM was also demonstrated on the UNB-ISCX dataset. The achieved accuracy was 96.28%. Gu et al. [17] proposed SKM-HFS, a semisupervised weighted k-mean framework using hybrid feature selection for DDoS attack detection. The Hadoop-based hybrid feature selection technique was used to determine the vital features to obtain higher detection accuracy. SKM-HFS experimented on the DARPA, CAIDA2007, and CICID2017 datasets with accuracies of 99.68%, 99%, and 98.86%, respectively. An experiment was conducted for a real-world dataset and achieved an accuracy of 99.75%. Bawany et al. [18] discussed major challenges and requirements for an effective DDoS mechanism. They proposed a customizable framework called ProDefense (SDN-Based Proactive DDoS Defense Framework) to defend against DDoS attacks. ProDefense is a scalable & lightweight application that provides rapid detection of DDoS attacks without utilizing high computation power. It supports blocking port, diverting flows, and controlling bandwidth techniques to mitigate DDoS attacks. Idhammad et al. [19] propose semi-supervised DDoS detection, combining an unsupervised ML technique and a supervised ensembled ML technique. The unsupervised approach is based on co-clustering, entropy estimation, and information gain ratio that helps it to remove noisy data from network traffic. The processed network traffic improves the performance and reduces the false-positive rate. The supervised ensemble ML classifiers classify the malicious traffic accurately and reduce the false-positive rates. Researchers experimented their approach on NSL-KDD, UNB ISCX IDS 2012, and UNSW-NB1 dataset getting accuracies of 98.23%, 99.88%, and 93.71%, respectively.

3 Proposed Framework

In this work, a combination of ML models are used to detect DDoS attacks. The systematic flow of VMFCVD is given here: Fig. 2 exhibits the same.

- The data cleaning module handles dataset noises such as single value columns, infinity values, and missing values well in advance.
- The data transformation module normalizes data so that all the features in the dataset will have a normalized data range.
- Extensive work on dimensionality reduction techniques ensures that all the features are ranked correctly based on various factors, such as feature importance, Pearson's correlation, and mutual information gain.
- The cluster formation module emphasizes the highly ranked features while creating clusters.
- The module to find the best cluster is designed precisely to rank all the clusters quickly.
- The ML models involved in voting are selected precisely. Each algorithm outperforms other algorithms one or more times during our testing on various datasets. There is no single algorithm that dominates others.
- The voting calculation is dynamic. The weight calculation module ensures that the high-performing algorithm obtains higher weight during voting.
- The prediction of the voting module for any network traffic is between 0 and 1; it is the probability of being benign or malicious flow. This value helps to fine-tune the final result of our voting classifier.
- All three modes are optimized based on their development detection strategy. FDM is optimized toward detection speed, DFDM is optimized toward detecting malicious flow with high speed, and HAM is optimized toward accuracy without concerning the detection speed.
- VMFCVD discards all the packets predicted as malicious to defend against a DDoS attack.
- VMFCVD is validated on most of the benchmark DDoS and botnet attack datasets.

3.1 Data Preprocessing

Datasets may have noise that may hamper the performance of a ML model. Table 7 shows that most of the datasets have NA values. NA values can reduce the performance of the model if not handled correctly. There are many datasets where one or more features have a unit value. Unit values are identical for all the records; therefore, they do not affect the model's accuracy, but they can increase the complexity of the model. Data transformation is equally essential to transform data from one dimension to another, convert nonnumeric data to numeric data or scale down the range of the dataset in a standard range of values. Data preprocessing includes all these steps and many more to produce a high-quality dataset so that any ML model can obtain a high accuracy while using low computational power. In VMFCVD, we worked extensively on data preprocessing, and the output of this module provides us with exceptionally high-quality dimensionally reduced noise-free datasets. Figure 3 exhibits an overview of our proposed data preprocessing module

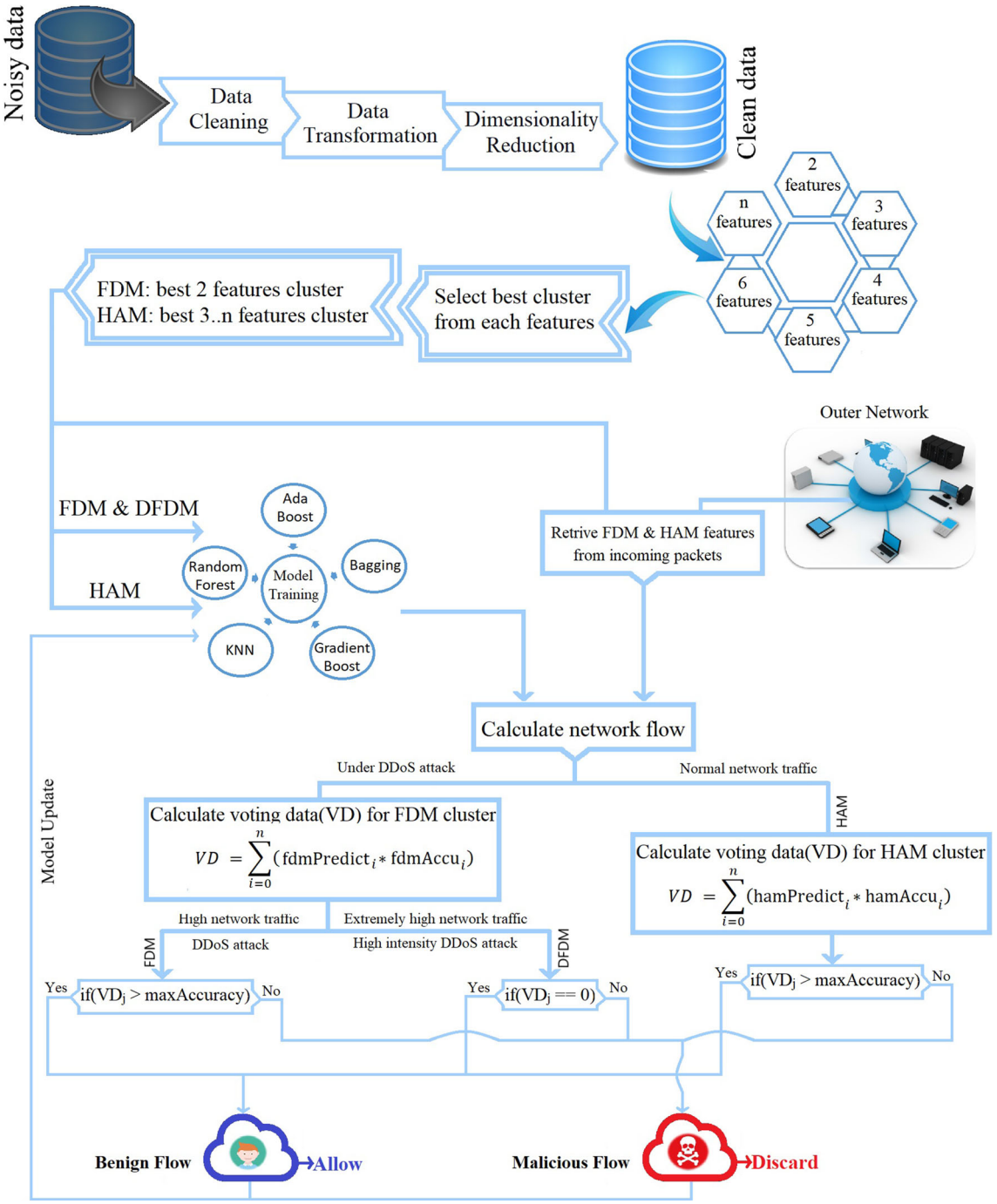

Fig. 2 Flow diagram of VMFCVD

3.1.1 Data Cleaning

Data cleaning is one of the crucial steps of data preprocessing. We started with deleting features that had identical values. Features with identical values for each record do not affect the accuracy of the ML model. In the second step of data cleaning, we replaced all the −inf and inf values with NA to handle them using our ImputeMissing module. It is required as −inf & inf values are not supported by many ML models. At the end of data cleaning, we called the ImputeMissing module to determine the most appropriate value for each missing value occurring in the dataset. Deleting these values can be another option to handle missing values, but deleting a single missing value deletes the entire row. This can lead to the loss of a considerable amount of essential data. For example, the original UNSW-NB15 dataset has 2.27% data missing, but deleting

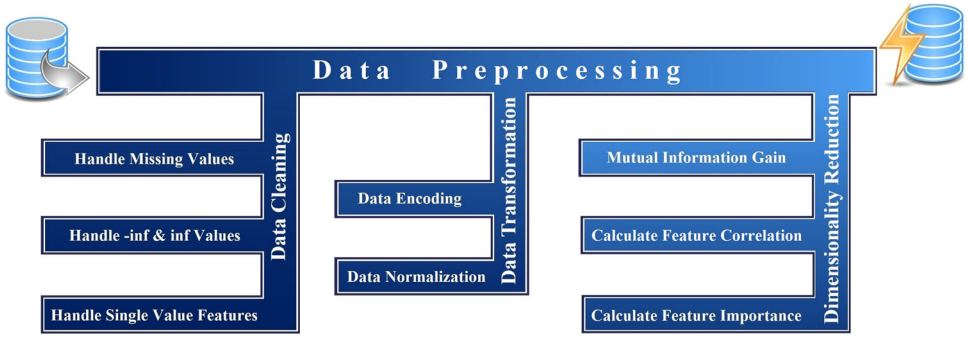

Fig. 3 Data preprocessing module

missing values deletes 47.8% of the dataset. During data preprocessing, if a dataset has more than 40% missing data for a particular feature, we delete that feature entirely. Including these features can slow down the detection process, as for each missing data, the ImputeMissing module needs to be called. Including features with missing values can increase the processing overhead for VMCFD. The impact is high when VMFCVD switches to the low feature-based detection module, where we consider a minimum number of features for attack detection to speed up the attack detection process.

The developed missing value imputation module uses a linear regression algorithm to predict the missing value. We first categorized the dataset into two categories: COLS _WONA, which consisted of features without missing values, and COLS_WNA, which consisted of the features with missing values. We predicted missing values for each feature of COLS_WNA. A train–test split was performed in such a way that Y_Train contained all the nonmissing values, and Y_Test had all the missing values. The linear regression model was trained using X_Train and Y_Train, and then, the Y_Test was predicted using X_Test data. The pseudo-code is given by Algorithm 1.

Algorithm 1 Missing value imputation Algorithm

Require: input dataset DS having NA values
Ensure: output dataset DS_WONA without NA values
1: *COLS_WONA* ⇐ Select columns in DS not having NA values
2: *COLS_WNA* ⇐ Select columns in DS having NA values
3: *DS_WONA* ⇐ DS (COLS_WONA)
4: **for** i=0 to len(COLS_WNA) **do**
5: *TRAIN* ⇐ Select all non NA rows of COLS_WNA[i] from DS
6: *TEST* ⇐ Select all rows from DS having NA value in COLS_WNA[i]
7: *X_Train* ⇐ TRAIN(COLS_WONA)
8: *Y_Train* ⇐ TRAIN(COLS_WNA[i])
9: model.fit(X_Train, Y_Train)
10: *X_Test* ⇐ *TEST*(*COLS_WONA*)
11: *Y_Test* ⇐ *model.predict*(*X_Test*)
12: *DS_WONA*[*COLS_WNA*[*i*]] ⇐ *Combine*(*Y_Train*, *Y_Test*)
13: **end for**

3.1.2 Data Transformation

This is a process to restructure the dataset from one to another. It enhances the dataset's quality and organizes it in a more precise format to improve the performance of ML models. Data normalization and data encoding are some resources involved in data transformation. Normalization is a process to rescale and transform the data so that each feature can have an identical range of values [20]. It ensures that the ML model is not biased toward any feature with a wide range of values compared to features with a value range in the single digits or a value even smaller than the single-digit range. It produces an improved version of the dataset with equal importance for each feature as input to the learning model. We used the StandardScaler transformation technique to normalize the dataset, where we scaled down all numeric values between −1 and 1. Data encoding is another imperative process used to transform categorical features into numerical features. Most ML models perform excellently on numerical features, while categorical features are not welcomed by many. Encoding categorical features to numeric features improves the performance of the model. We have applied OneHotEncoder, the most widely used data encoding technique, to transform categorical features into numerical features.

3.1.3 Dimensionality Reduction

Dimensionality reduction is a technique to identify an extraneous feature of a dataset that either decreases the ML model's performance or does not improve the ML model's performance. Datasets may have redundant features and irrelevant information, negatively affecting the ML model's performance [21]. Figures 4 and 5 show that if we increase the number of features, it increases the time complexity of the model. It is also observed that increasing the feature size does not mean that it will increase the model's accuracy. It is essential to achieve a reduced set of features that can give high accuracy and reduce the difficulties of the learning process [22]. Our investigation shows that lowering the dimension improves the detection speed. We have considered feature

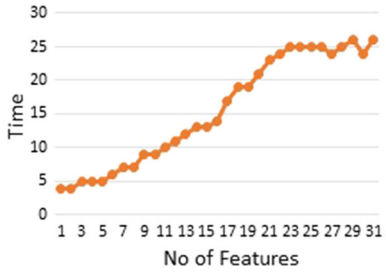

Fig. 4 Increase in complexity

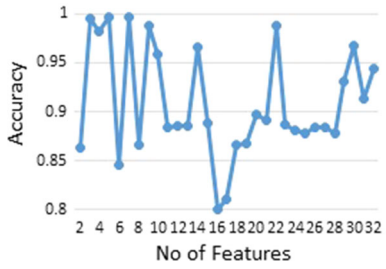

Fig. 5 Unpredictable accuracy

importance (FI), mutual information gain, and feature correlation to obtain high-ranked features for training the voting model.

Feature Importance Feature importance describes how important a feature is for the model's classification performance. There are different ML models that help to measure feature importance. During the experiment, it was observed that different ML models give diverse feature importance (FI) values on the same dataset. Each ML model's FI values were different; for the same feature, one algorithm showed zero importance, albeit the other showed a considerable positive value. To ensure that we did not miss any potential features, we consolidated three ML models, namely LightGBM, XGBoost, and DecessionTree, to measure feature importance.

We observed that the data range of each ML model's FI varied widely. After calculating FI, we rescaled the FI range for each algorithm to the range of [0, 1]. The min–max normalization function is given as:

$$\bar{X} = \frac{X - min(X)}{max(X) - min(X)} \tag{1}$$

where

X is a list of features with their calculated FI
$\bar{X}$ is the normalized value calculated from X.

Once the FI values were measured and normalized, we combined the FI values of all three models. A graph is plotted in Fig. 6 based on the combined FI values. Features with higher FI values are essential features for the classification model.

Feature Correlation This is the crucial stage where we determined the correlation between all the features in the dataset. If two features are highly correlated, then we can keep one feature and drop the other feature. Features that are highly correlated with classification output are preferred; features highly associated with other features are not suitable for ML algorithms [23]. We calculated the correlation between features using Pearson's correlation coefficient formula.

$$P_{XY} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \tag{2}$$

where

X and Y are two features of the dataset
n is the number of rows in X
$\bar{X}$ is mean of X
$\bar{Y}$ is mean of Y
P_{XY} is Pearson's correlation coefficient between X and Y.

A heatmap is plotted in Fig. 7 based on the calculated correlation values between different features. The correlation map shows how much two features are linearly correlated. The possible range of values for the correlation coefficient is -1 to 1. A value equal to zero or close to zero indicates that there is no relation between two features. A value equal to 1 or -1 or close to it indicates that both features are highly correlated.

Mutual Information Mutual information is calculated between an independent feature and a dependent feature to determine the information gain value, which measures the dependency of a dependent feature on an independent feature. Features with a higher value of information gain can be included in the minimal set of features on the ML algorithm. We used Mutual_info_classif to determine the MI value for each feature concerning the classification output. Figure 8 depicts the graph plotted from the calculated mutual information gain. The range of MI values can be between 0 and 1. A high MI score of a feature means it has a closer connection with the target feature and including it in the training set can be helpful for the classification model.

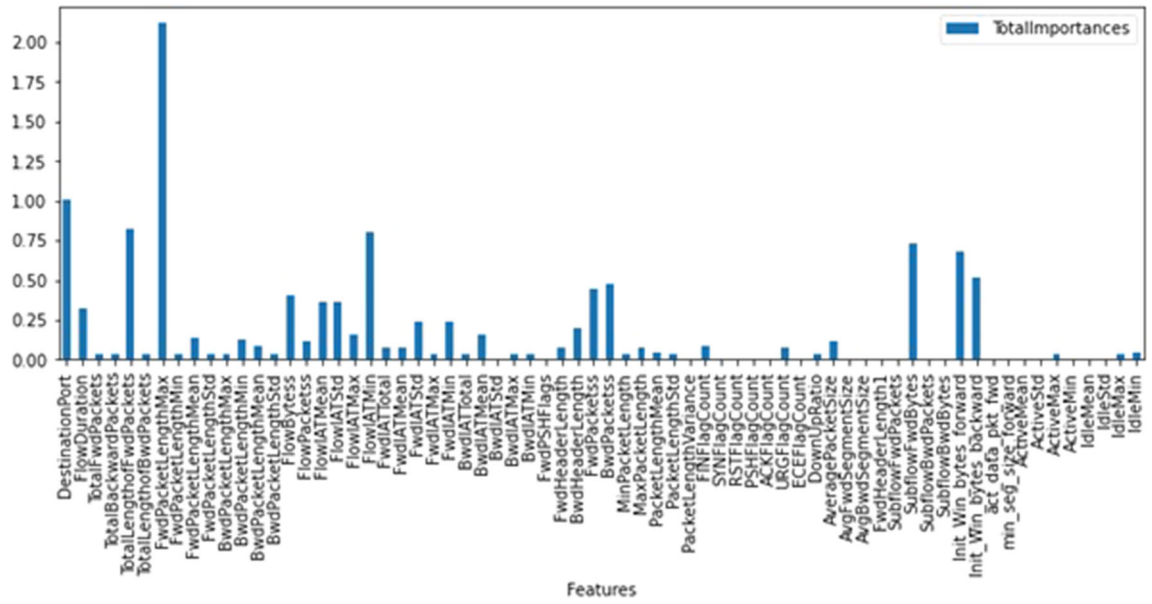

Fig. 6 Combined feature importance

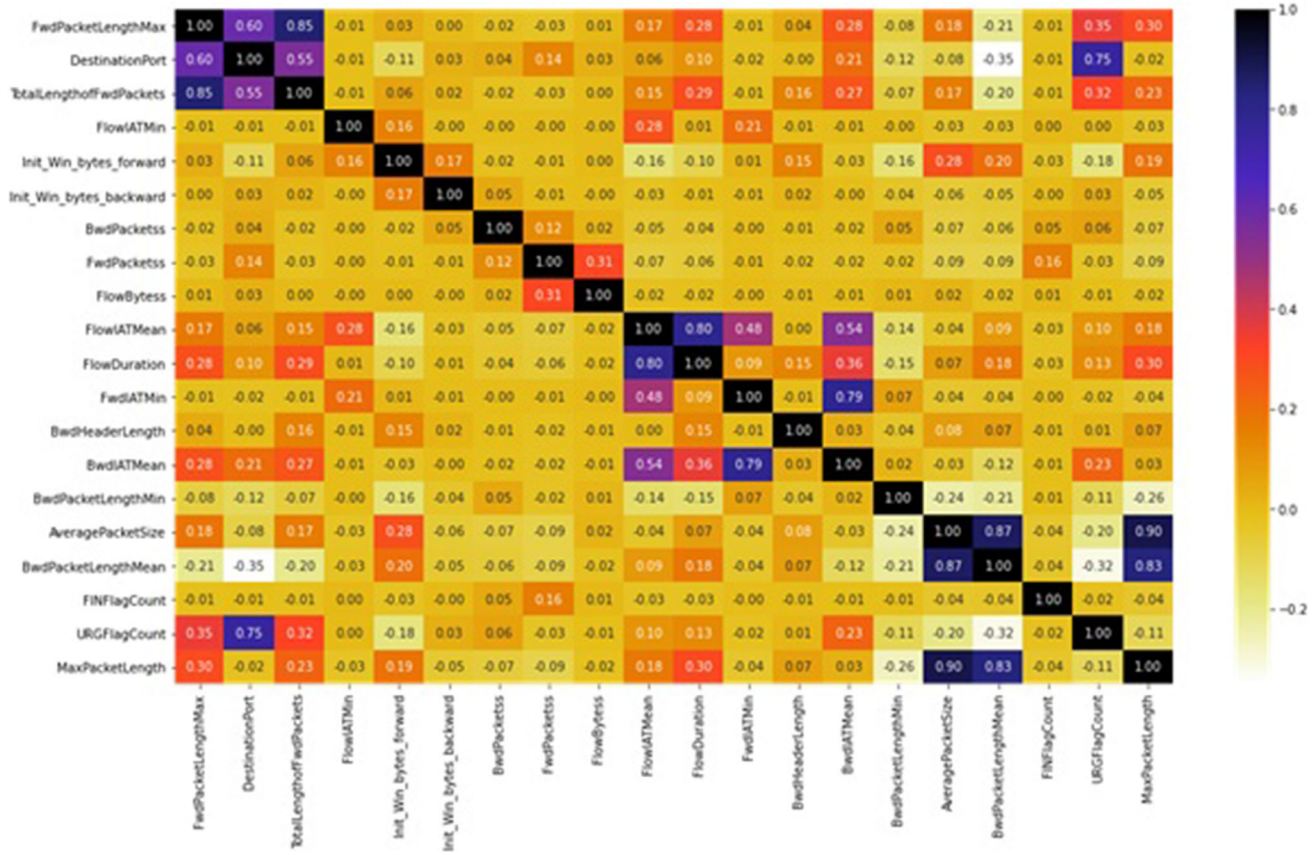

Fig. 7 Heatmap based on features correlation

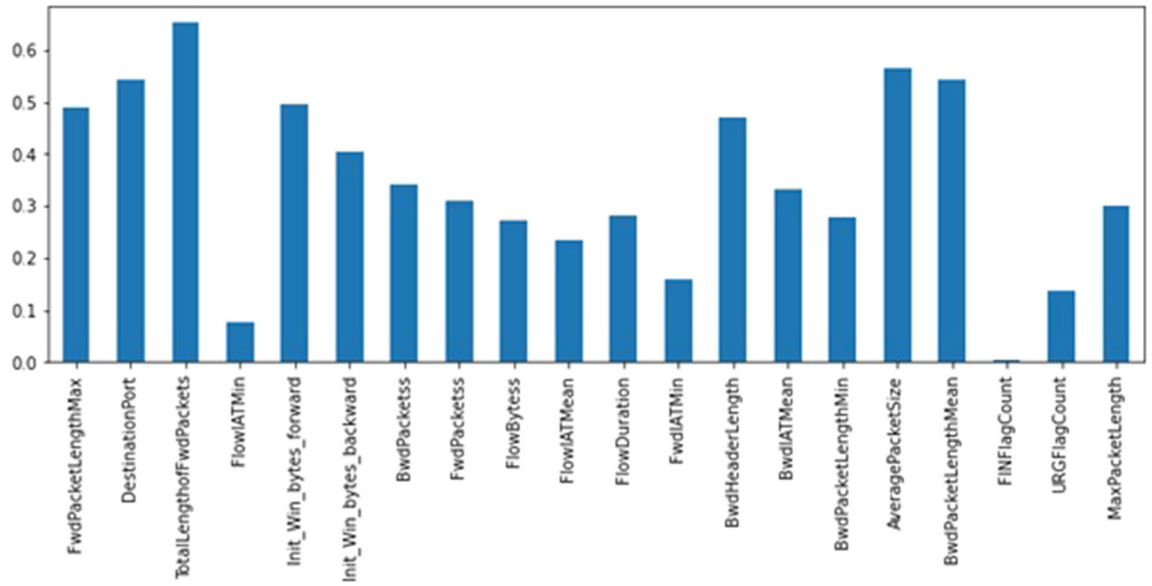

Fig. 8 Graph plotted from mutual information gain

Algorithm 2 ClusterFormation algorithm

Require: FEATURES: list of features in dataset
FeaturesInEachCluester: no. of features in each subcluster
NoOfFeatures: no of features considered while creating subclusters
Ensure: CLUSTERS: final set of clusters
1: *CLUSTERS* ⇐ []
2: **if** *NoOfFeatures* < 3 **then**
3: *NoOfFeatures* ⇐ 3
4: **end if**
5: **for** i=0 to NoOfFeatures **do**
6: **for** j=i+1 to NoOfFeatures **do**
7: *NewRow* ⇐ []
8: NewRow.append(FEATURES[i])
9: **for** k=0 to FeaturesInEachCluester **do**
10: NewRow.append(FEATURES [j+k])
11: **end for**
12: CLUSTERS.append(NewRow)
13: **end for**
14: **end for**

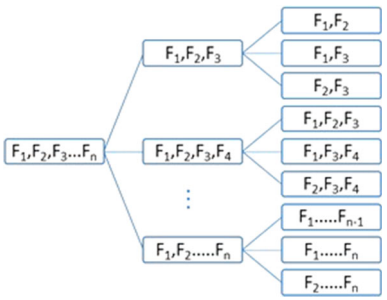

Fig. 9 Data preprocessing module

3.2 Cluster Formation, Ranking, and Best Cluster Identification

Once the features were ranked from the techniques explained above, we needed to determine the minimal features from these selected features. The output of the data preprocessing and feature selection phase was a list of features with their ranking from highest to lowest. The highest-ranked features had a high potential for the minimal set of features for our detection framework. It is essential to identify high-ranked features that are good together to improve the model's accuracy. In the next step, we clustered the datasets so that highly ranked features obtained more weight in the clustering process. The pseudo-code is given by Algorithm 2. The output of ClusterFormation is represented in Fig. 9 when n features were called one by one. We call ClusterFormation module $n-2$ times where n is total number of features. First, we call it for three features, then for four features, and so on until it reaches the last feature. Each time when we call ClusterFormation, it generates three subclusters. The accuracy of all these subclusters is calculated in next stage, and we select the best-performing cluster only.

We calculated the accuracy of each subcluster and selected the best-performing cluster for the next step. Table 1 shows details of the best cluster from the set of subclusters.

The best-performing cluster with two features was selected for one of the modes of VMFCVD. For another mode, we chose the best-performing cluster from the rest of the cluster sets. Table 2 gives information about the fastest cluster and cluster that can give the highest accuracy.

Table 1 Best clusters from each subcluster

NoOfFeatures	Feature set	Accuracy
2	[F53, F1]	0.9886
3	[F53, F13, F1]	0.9956
4	[F53, F13, F1, F67]	0.9813
5	[F5, F53, F13, F1, F67]	0.8954
6	[F53, F13, F1, F67, F7, F36]	0.9963
7	[F53, F13, F1, F67, F7, F36, …	0.8166
8	[F53, F13, F1, F67, F7, F36, …	0.8166
9	[F53, F13, F1, F67, F7, F36, …	0.8166
10	[F53, F13, F1, F67, F7, F36, …	0.7154
11	[F5, F13, F1, F67, F7, F36, …	0.7175
12	[F53, F13, F1, F67, F7, F36, …	0.9464
13	[F53, F13, F1, F67, F7, F36, …	0.9473
14	[F53, F13, F1, F67, F7, F36, …	0.9473
15	[F5, F53, F13, F1, F67, F7, …	0.8348
16	[F53, F13, F1, F67, F7, F36, …	0.9669
17	[F53, F13, F1, F67, F7, F36, …	0.969
18	[F53, F13, F1, F67, F7, F36, …	0.9799

3.3 Voting-based Volumetric DDoS Detection Framework

Factors such as dataset size and deployment environment extensively affect ML model performance. It is not advisable to depend on a single ML algorithm when developing a framework. Although a ML algorithm can perform exceptionally well on a particular dataset with high accuracy and precision, there is no guarantee that it will perform well on all datasets [24]. VMFCVD relies on the triad of fast detection mode (FDM), defensive fast detection mode (DFDM), and high accuracy mode (HAM) models. The implementation of this triad ensures that the system will defend itself against DDoS attacks more efficiently. We considered five well-known ML models to classify network traffic: AdaBoost classifier, bagging classifier, gradient boosting classifier, K-neighbors classifier, and random forest classifier. The result was calculated based on their voting. Each ML algorithm has a different voting right based on its performance calculated during its training time. VMFCVD is highly adaptive. It has three different approaches to classify cyber-attacks and switches between these approaches based on the network traffic.

Algorithm 3 Voting Calculation Algorithm

Require: MODELS: [List of ML models]
Ensure: Final calculated voting
1: $VotingData \Leftarrow 0$, $VoteIndex \Leftarrow 0$, $TotAccuracy \Leftarrow 0$
2: **if** $NoOfFeatures < 3$ **then**
3: $model \Leftarrow model.fit(X\_Train, Y\_Train)$
4: **end if**
5: **for** model in MODELS **do**
6: $model \Leftarrow model.fit(X\_Train, Y\_Train)$
7: $predict \Leftarrow model.predict(X\_Test)$
8: $accuracy \Leftarrow metrics.accuracy\_score(Y\_Test, predict)$
9: $VotingData \Leftarrow VotingData + predict * accuracy$
10: $TotAccuracy \Leftarrow TotAccuracy + accuracy$
11: **end for**
12: $VoteIndex \Leftarrow TotAccuracy/2 - 0.2$
13: $MaxVoteIndex \Leftarrow 0$
14: $TotalRecords \Leftarrow Len(X\_Test)$
15: **for** i=0 to 40 **do**
16: **if** $VotingData >= VoteIndex$ **then**
17: $Vote \Leftarrow 1$
18: **else**
19: $Vote \Leftarrow 0$
20: **end if**
21: $Accuracy \Leftarrow Sum(Vote)/TotalRecords$
22: **if** $MaxAccuracy < Accuracy$ **then**
23: $MaxAccuracy \Leftarrow Accuracy$
24: $MaxVoteIndex \Leftarrow VoteIndex$
25: **end if**
26: $VoteIndex \Leftarrow VoteIndex + 0.01$
27: **end for**
28: $Accuracy \Leftarrow Sum(Vote)/TotalRecords$
29: **if** Y_Test=1 AND Vote=1 **then**
30: $TP \Leftarrow 1$
31: **else**
32: $TP \Leftarrow 0$
33: **end if**
34: **if** Y_Test=0 AND Vote=0 **then**
35: $TN \Leftarrow 1$
36: **else**
37: $TN \Leftarrow 0$
38: **end if**
39: **if** Y_Test=0 AND Vote=1 **then**
40: $FP \Leftarrow 1$
41: **else**
42: $FP \Leftarrow 0$
43: **end if**
44: **if** Y_Test=1 AND Vote=0 **then**
45: $FN \Leftarrow 1$
46: **else**
47: $FN \Leftarrow 0$
48: **end if**

Calculation of Vote All the ML models were trained on the training dataset. Once trained, the accuracy and their predictions were calculated on the test dataset. We calculated the weight of each ML algorithm based on its accuracy. Once

Table 2 Fastest and highest performing clusters

NoOfFeatures	Feature set	Accuracy	Conclusion
2	[F53, F1]	0.9886	Fastest cluster
6	[F53, F13, F1, F67, F7, F36]	0.9963	Most accurate cluster

Table 3 Data reduction in FDM

Dataset	Dimensionality reduction		FDM Data reduction (%)
	Before	After	
CICIDS2017 BoT	79	2	97.4
CICIDS2017 DDoS	79	2	97.4
CSE-CIC-IDS2018 BoT	80	2	97.5
CSE-CIC-IDS2018 DDoS	80	2	97.5
CICDDoS2019 DNS	88	2	97.7
CICDDoS2019 LDAP	88	2	97.7
CICDDoS2019 SSDP	88	2	97.7
CICDDoS2019 SYN	88	2	97.7
DoHBrw 2020	35	2	94.2
NBaIoT2018 Mirai	116	2	98.2
UNSW NB15	46	2	95.8
UNSW2018 BoTIoT	48	2	95.6

Table 4 Improvement in DFDM over FDM

Dataset	Accuracy		
	FDM	DFDM	Improvement
CICIDS2017 BoT	0.273187	0.284992	0.0118
CICIDS2017 DDoS	0.998191	0.9984	0.00021
CSE-CIC-IDS2018 BoT	0.989981	0.993876	0.00389
CSE-CIC-IDS2018 DDoS	0.997507	1	0.00249
CICDDoS2019 DNS	0.999988	0.99999	0
CICDDoS2019 LDAP	0.999947	1	0.00005
CICDDoS2019 SSDP	0.999999	0.999999	0
CICDDoS2019 SYN	0.999983	0.999998	0.00001
DoHBrw 2020	0.999987	0.999987	0
NBaIoT2018 Mirai	0.999931	0.999969	0.00004
UNSW NB15	0.996412	1	0.00359
UNSW2018 BoTIoT	0.999987	0.999995	0.00001

the weight was calculated, we multiplied the prediction of each ML algorithm by its weight for each record for every ML model's prediction. The resultant weighted prediction was used to calculate the voting for a packet to determine whether it was benign or malicious. Algorithm 3 is a master algorithm to perform the vote. All three modes of VMFCVD call it and perform the voting based on their voting criteria.

3.3.1 Fast Detection Mode (FDM)

As shown in Fig. 4, if we increased the feature size, the training time also increased, and the framework used more computational power. Considering this, we trained FDM with the minimum possible number of features set. Taking only two features from the network traffic enables FDM to classify more network traffic. FDM used a highly dimensionally reduced dataset provided by our dimensionality reduction module. Table 3 shows the percentage of data reduction for all the datasets we used during the experiment. Keeping the minimum number of features in the training and test datasets helped VMFCVD process the maximum number of requests compared to the mode where we considered more features for model training and testing. FDM was active all the time, even when HAM was running. During this time, FDM trained itself with all the incoming flows. VMFCVD automatically switched to FDM when it observes considerably high network traffic. HAM suspends during this time. Suspending HAM helps FDM obtain the maximum amount of resources for attack detection.

3.3.2 Defensive Fast Detection Mode (DFDM)

This is the extended version of the FDM. DFDM uses FDM's voting module, but it has voting calculation techniques. Even if a single vote is against a packet, that packet is classified as a malicious packet. Only the packets that have zero prob-

Table 5 Accuracy improvement in HAM over FDM

Dataset	Accuracy		
	FDM	HAM	Improvement
CICIDS2017 BoT	0.99207	0.99721	0.00514
CICIDS2017 DDoS	0.99855	0.99923	0.00068
CSE-CIC-IDS2018 BoT	0.99555	0.99555	0
CSE-CIC-IDS2018 DDoS	0.99829	0.99997	0.00168
CICDDoS2019 DNS	0.99997	0.99998	0.00001
CICDDoS2019 LDAP	0.99984	0.99999	0.00015
CICDDoS2019 SSDP	0.99991	0.99999	0.00008
CICDDoS2019 SYN	0.99994	0.99999	0.00005
DoHBrw 2020	0.99998	0.99998	0
NBaIoT2018 Mirai	0.99993	0.99995	0.00002
UNSW NB15	0.99633	0.99641	0.00008
UNSW2018 BoTIoT	0.99996	1	0.00004

ability of being malicious are classified as benign packets. This mode activates when the network experiences an attack with a very high volume of incoming data. DFDM increases the accuracy of detecting malicious packets but decreases the accuracy for detecting benign packets. VMFCVD switches to DFDM mode when the DDoS attack intensity is very high. The increase in accuracy of detecting malicious packets in DFDM over FDM is given in Table 4.

3.3.3 High Accuracy Mode (HAM)

This mode gives the highest accuracy. HAM may require a higher number of input features for training, which increases computational overhead. This mode activates when the network is stable, and the number of incoming packets is average. This model is intended to give the highest accuracy on any dataset. Table 5 shows the improvement in HAM's accuracy over FDM. The cluster that outperformed the others during feature selection time with the highest accuracy is considered input for this model.

4 Experimental Results and Discussion

All experiments are carried out on a 64-bit Window 10 Pro operating system equipped with a 2.70 GHz Core i7 processor, 16 GB RAM. VMFCVD is implemented in Python 3.8.8 using Jupyter Notebook. The machine learning classification models used are the AdaBoost classifier, bagging classifier, gradient boosting classifier, K-neighbors classifier, and random forest classifier. These classifiers are selected based on their diverse performance on the various datasets. Table 6 shows the diversity in their performance; if an ML model gives the highest accuracy for one dataset, the same model gives the lowest accuracy for another. The cells highlighted show the maximum performance of an ML model. AdaBoost gives the highest average performance, while bagging gives the best performance on five occasions. The lowest performance of bagging and GradientBoost is recorded three times.

4.1 Benchmark Datasets

Our framework has experimented with twelve labeled datasets containing benign and malicious network traffic. Most of the datasets we have considered have been generated

Table 6 Performance of ML models on various datasets

Dataset	Features	Accuracy				
		AdaBoost	Bagging	GB[a]	KNN	RF
CICIDS2017 BoT	2	**0.99194**	0.98696	0.98892	0.99082	0.98926
CICIDS2017 DDoS	2	**0.99838**	0.79279	0.89456	0.89815	0.79911
CSE-CIC-IDS2018 BoT	2	0.99199	0.98656	0.98815	**0.99537**	0.98318
CSE-CIC-IDS2018 DDoS	2	0.99823	0.99824	0.99824	**0.99828**	0.99824
CICDDoS2019 DNS	2	0.99986	0.99996	0.99995	**0.99997**	**0.99997**
CICDDoS2019 LDAP	2	0.99983	0.99983	0.99981	**0.99984**	0.99983
CICDDoS2019 SSDP	2	0.99989	**0.99991**	0.99981	0.99985	0.99989
CICDDoS2019 SYN	2	0.99979	**0.99993**	0.99978	**0.99993**	0.99978
DoHBrw 2020	2	0.99993	**0.99997**	**0.99997**	0.99996	0.99542
NBaIoT2018 Mirai	2	**0.99993**	**0.99993**	0.99991	**0.99993**	**0.99993**
UNSW NB15	2	0.99632	**0.99633**	0.99632	0.99287	0.99632
UNSW2018 BoTIoT	2	0.99989	**0.99996**	0.99988	0.99995	0.99991
Average	2	**0.99782**	0.97821	0.98777	0.98863	0.97827

The bold values highlights the best performing model's accuracy on a particular dataset. If more than one values are bold for a dataset, more than one model achieves the highest accuracy

[a]Gradient Boosting Classifier

Table 7 Detailed information about the datasets used for experiment

Dataset	Records	Features	Benign (%)	Malicious (%)	NA[a]	UnitValue[b]
CICIDS2017 BoT	191,033	79	98.97	1.029	244	10
CICIDS2017 DDoS	225,745	79	43.287	56.713	68	10
CSE-CIC-IDS2018 BoT	1,048,575	80	72.707	27.293	8100	10
CSE-CIC-IDS2018 DDoS	1,048,575	80	34.412	65.588	0	10
CICIDS2019 DNS	5,074,413	88	0.067	99.933	324,788	12
CICDDoS2019 LDAP	2,181,542	88	0.074	99.926	77,300	14
CICDDoS2019 SSDP	2,611,374	88	0.029	99.971	84,100	12
CICIDS2019 SYN	1,582,681	88	0.025	99.975	404,634	12
DoHBrw2020	269,643	35	7.346	92.654	688	0
NBaIoT2018 Mirai	498,164	116	12.477	87.523	0	0
UNSW2018 BoTIoT	3,668,522	46	0.013	99.987	0	0
UNSW NB15	2,540,047	48	12.649	87.351	2,778,024	0

[a]Total count of NA values
[a]Total number of columns with a single value

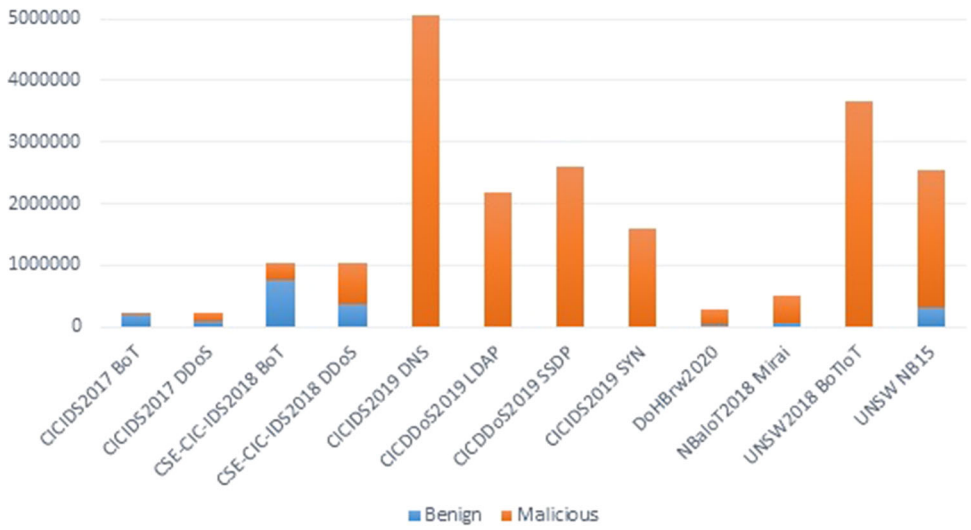

Fig. 10 Distribution of benign and malicious records

in recent years and include new kinds of attacks. Table 7 gives detailed information on all the datasets used in this work. For DDoS attack detection, we considered CICIDS2017 DDoS, CSE-CIC-IDS2018 DDoS, and CICDDOS2019 DDoS. The CICIDS2017 BoT, CSE-CIC-IDS2018 BoT, NBaIoT2018 Mirai, UNSW2018 BoTIoT, and UNSW NB15 datasets were considered for the botnet attack. The CICIDS 2017 BoT and CICIDS 2017 DDoS dataset are subsets of the CICIDS2017 [25] dataset created by the Canadian Institute for Cybersecurity. The CICIDS 2017 BoT has 99% benign and 1% malicious flows, while the CICIDS 2017 DDoS has 43% benign and 57% malicious flows. The CSE-CIC-IDS2018 BoT and DDoS are subsets of the CSE-CIC-IDS2018 [25] dataset. CSE-CIC-IDS2018 is an updated version of the CICIDS2017 dataset. We considered the DNS, LDAP, SSDP, and SYN datasets, a subset of the CICDDOS2019 [26] dataset. CICDDoS2019 has a large number of malicious records with only a few benign records. CICDDoS2019 DNS and 0.067% benign records, CICDDoS2019 LDAP have 0.074%, CICDDoS2019 SSDP have 0.029% and CICDDoS2019 SYN have only 0.025% benign records.

DoHBrw-2020 is the most recent dataset generated by the Canadian Institute for Cybersecurity. It contains a hybrid of modern benign and malicious network traffic [27]. NBaIoT2018 Mirai is a subset of the NBaIoT2018 dataset collected from nine infected IoT devices [28]. UNSW2018 BoTIoT is extracted from the UNSW BoTIoT dataset that incorporates legitimate IoT network traffic [29]. The UNSW-NB15 dataset is a well-structured dataset to evaluate cyber-attack detection systems created at the University of New South Wales in 2015 [30]. Figure 10 depicts the distribution of benign vs. malicious records.

Table 8 Comparison between FDM, DFDM, and HAM

Dataset	Accuracy			Precision			Sensitivity			F1Score		
	FDM	DFDM	HAM	FDM	DFDM	HAM	FDM	DFDM	HAM	FDM	DFDM	HAM
CICIDS2017 BoT	0.9921	0.9905	0.9972	0.9922	0.9923	0.9981	0.9999	0.9982	0.9991	0.996	0.9952	0.9986
CICIDS2017 DDoS	0.9986	0.9982	0.9992	0.9976	0.9979	0.9984	0.999	0.998	0.9998	0.9983	0.9979	0.9991
CSE-CIC-IDS2018 BoT	0.9956	0.9927	0.9956	0.9963	0.9977	0.9963	0.9976	0.9922	0.9976	0.997	0.9949	0.997
CSE-CIC-IDS2018 DDoS	0.9983	0.9982	0.9999	0.9954	0.9999	0.9999	0.9998	0.995	0.9999	0.9976	0.9975	0.9999
CICDDoS2019 DNS	0.9999	0.9999	0.9999	0.9818	0.985	0.9869	0.9809	0.9274	0.9869	0.9813	0.9554	0.9869
CICDDoS2019 LDAP	0.9998	0.9992	0.9999	0.9248		0.9999	0.8531		0.9878	0.8875		0.9938
CICDDoS2019 SSDP	0.9999	0.9999	0.9999	0.9935	0.9924	0.9817	0.7097	0.5991	0.9862	0.828	0.7471	0.9839
CICDDoS2019 SYN	0.9999	0.9998	0.9999	0.934	0.96	0.9999	0.839	0.2034	0.9746	0.8839	0.3357	0.9871
DoHBrw 2020	0.9999	0.9999	0.9999	0.9998	0.9998	0.9998	0.9999	0.9993	0.9999	0.9999	0.9996	0.9999
NBaIoT2018 Mirai	0.9999	0.9999	0.9999	0.9995	0.9998	0.9996	0.9999	0.9997	0.9999	0.9997	0.9997	0.9998
UNSW NB15	0.9963	0.9871	0.9964	0.7836		0.7841	0.9904		0.9982	0.8749		0.8783
UNSW2018 BoTIoT	0.9999	0.9999	0.9999	0.8772	0.7222	0.9999	0.7692	0.1	0.9999	0.8197		0.9999

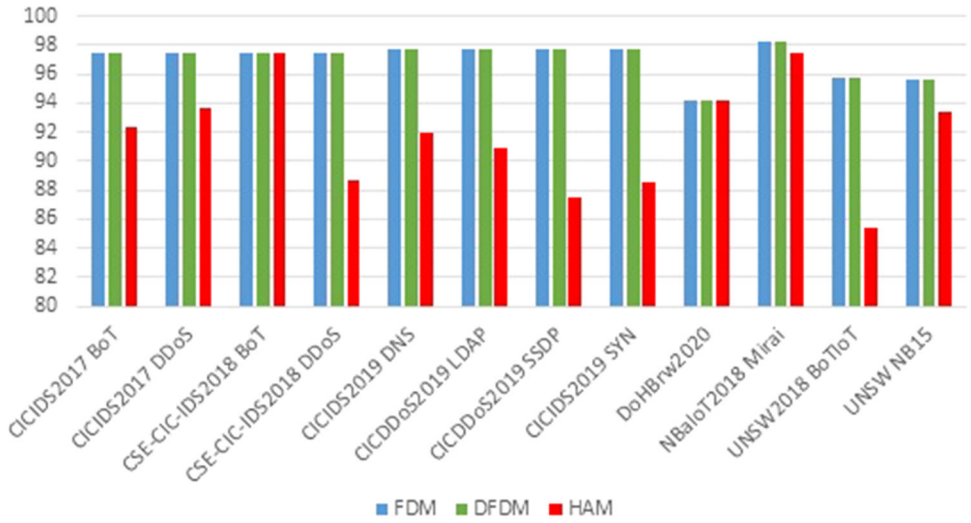

Fig. 11 Dimensionality reduction in FDM, DFDM, and HAM

Table 9 Accuracy of FDM over ML models

Dataset	AdaBoost	Bagging	GB	KNN	RF	VMFCVD
CICIDS2017 BoT	0.99194	0.98696	0.98892	0.99082	0.98926	0.99207
CICIDS2017 DDoS	0.99838	0.79279	0.89456	0.89815	0.79911	0.99855
CSE-CIC-IDS2018 BoT	0.99199	0.98656	0.98815	0.99537	0.98318	0.99555
CSE-CIC-IDS2018 DDoS	0.99823	0.99824	0.99824	0.99828	0.99824	0.99829
CICDDoS2019 DNS	0.99986	0.99996	0.99995	0.99997	0.99997	0.99997
CICDDoS2019 LDAP	0.99983	0.99983	0.99981	0.99984	0.99983	0.99984
CICDDoS2019 SSDP	0.99989	0.99991	0.99981	0.99985	0.99989	0.99991
CICDDoS2019 SYN	0.99979	0.99993	0.99978	0.99993	0.99978	0.99994
DoHBrw 2020	0.99993	0.99997	0.99997	0.99996	0.99542	0.99998
NBaIoT2018 Mirai	0.99993	0.99993	0.99991	0.99993	0.99993	0.99993
UNSW NB15	0.99632	0.99633	0.99632	0.99287	0.99632	0.99633
UNSW2018 BoTIoT	0.99989	0.99996	0.99988	0.99995	0.99991	0.99996

4.2 Evaluation Metrics

All three detection modes of VMFCVD are evaluated based on the four evaluation metrics, viz. accuracy, precision, sensitivity, and F1 score. The equation for the same is given below:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + FP} \tag{3}$$

$$Precision = \frac{TP}{TP + FP} \tag{4}$$

$$Sensitivity(Recall) = \frac{TP}{TP + FN} \tag{5}$$

$$F1\,Score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{6}$$

4.3 Experimental Result

The analysis of the experimental results is divided into three sections. In the first section, we will compare all three modes of VMFCVD. In the second section, VMFCVD is compared with ML models' performance. In the last section, we will compare the performance of VMFCVD with state-of-the-art baselines.

4.3.1 Performance Analysis Between FDM, DFDM, and HAM

These three modes are designed to detect an attack in different scenarios that primarily affect these modes' performance. HAM outperformed FDM and DFDM when comparing accuracy, sensitivity, and F1 score. Considering the extensive dimensionality reduction in FDM, the accuracy of FDM was quite competitive to HAM. The precision of DFDM outperformed FDM for all datasets and topped HAM in many cases. The highlighted cells in Table 8 show the outperformed values.

Figure 11 shows the dimensionality reduction in various datasets. The minimum dimensionality reduction is 85.4 on the UNSW NB15 dataset for HAM. The maximum dimensionality reduction we achieved was 98.2%, which was on

Table 10 Accuracy of DFDM over ML models for detecting malicious packets

Dataset	AdaBoost	Bagging	GB	KNN	RF	DFDM
CICIDS2017 BoT	0.00169	0.33895	0	0.82462	0	0.28499
CICIDS2017 DDoS	0.99824	0.63935	0.81557	0.82194	0.64761	0.9984
CSE-CIC-IDS2018 BoT	1	1	1	0.99751	1	0.99388
CSE-CIC-IDS2018 DDoS	0.99062	0.9525	0.96161	0.98858	0.95228	1
CICDDoS2019 DNS	0.99993	0.999981	0.99998	0.99998	0.99998	0.99999
CICDDoS2019 LDAP	0.99993	0.99994	0.99981	0.99994	0.99994	1
CICDDoS2019 SSDP	1	0.52123	0.99983	0.99982	1	1
CICDDoS2019 SYN	0.99998	0.99997	0.99998	0.99998	0.9999	1
DoHBrw 2020	0.99997	0.99999	0.99997	0.99997	0.99995	0.99999
NBaIoT2018 Mirai	0.99993	0.99993	0.99991	0.99992	0.99997	0.99997
UNSW NB15	0.98527	0.98526	0.98526	0.98527	0.98527	1
UNSW2018 BoTIoT	1	0.99999	0.99949	1	1	1

Table 11 Accuracy of HAM over ML models

Dataset	AdaBoost	Bagging	GB	KNN	RF	HAM
CICIDS2017 BoT	0.98928	0.99263	0.98919	0.99721	0.98926	0.99721
CICIDS2017 DDoS	0.43027	0.99913	0.99875	0.99904	0.99913	0.99923
CSE-CIC-IDS2018 BoT	0.99199	0.98656	0.98815	0.99537	0.98318	0.99555
CSE-CIC-IDS2018 DDoS	0.99996	0.99996	0.35197	0.99991	0.71474	0.99997
CICDDoS2019 DNS	0.99983	0.9999	0.99925	0.99997	0.99994	0.99998
CICDDoS2019 LDAP	0.99998	0.99999	0.99991	0.99998	0.99998	0.99999
CICDDoS2019 SSDP	0.99996	0.99999	0.99992	0.99997	0.99998	0.99999
CICDDoS2019 SYN	0.99999	0.99999	0.99987	0.99999	0.99999	0.99999
DoHBrw 2020	0.99993	0.99997	0.99997	0.99996	0.99542	0.99998
NBaIoT2018 Mirai	0.99993	0.99995	0.99995	0.99995	0.99994	0.99995
UNSW NB15	0.99003	0.9963	0.9902	0.99635	0.99313	0.99641
UNSW2018 BoTIoT	1	1	1	1	1	1

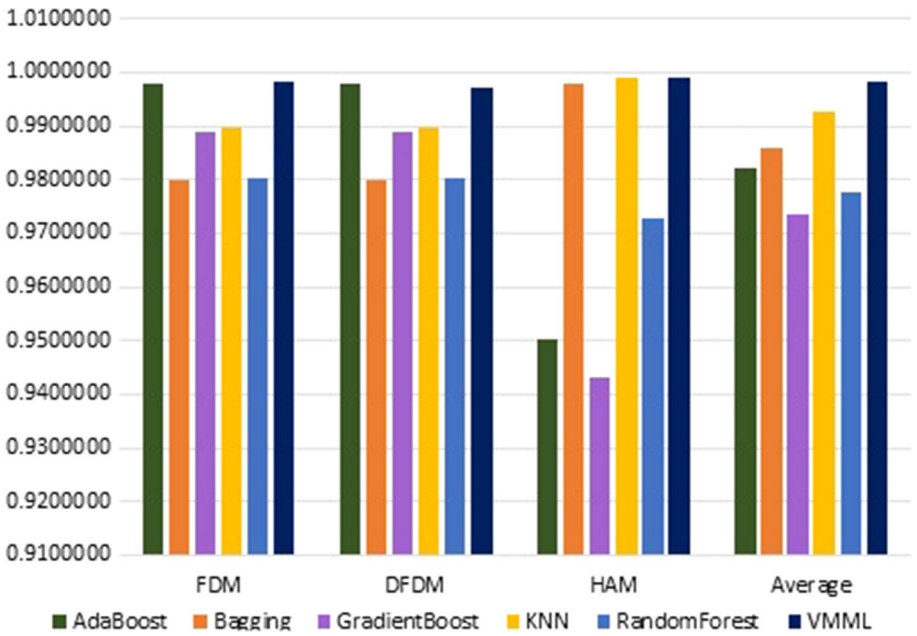

Fig. 12 Accuracy (average) of VMFCVD over ML models

Table 12 Accuracy of VMFCVD over state-of-the-art baselines

Dataset	State of the art		VMFCVD Accuracy		
	Reference	Accuracy	FDM	DFDM	HAM
CICIDS2017 BoT	[31]	0.46474	0.99207	0.99053	**0.99721**
CICIDS2017 BoT	[32]	0.8719	0.99207	0.99053	**0.99721**
CICIDS2017 DDoS	[9]	0.9967	0.99855	0.99822	**0.99923**
CICIDS2017 DDoS	[13]	0.982	0.99855	0.99822	**0.99923**
CICIDS2017 DDoS	[21]	0.82	0.99855	0.99822	**0.99923**
CICIDS2017 DDoS	[31]	0.99879	0.99855	0.99822	**0.99923**
CSE-CIC-IDS2018 BoT	[33]	**0.9992**	0.99555	0.99265	0.99555
CSE-CIC-IDS2018 DDoS	[9]	0.9987	0.99829	0.99823	**0.99997**
CICDDoS2019 DNS	[14]	0.9975	0.99997	0.99994	**0.99998**
CICDDoS2019 LDAP	[14]	0.9996	0.99984	0.99924	**0.99999**
CICDDoS2019 LDAP	[13]	0.953	0.99984	0.99924	**0.99999**
CICDDoS2019 LDAP	[32]	0.9995	0.99984	0.99924	**0.99999**
CICDDoS2019 SSDP	[14]	0.9991	0.99991	0.99988	**0.99999**
CICDDoS2019 SSDP	[34]	0.86	0.99991	0.99988	**0.99999**
CICDDoS2019 SYN	[14]	0.9998	0.99994	0.99977	**0.99999**
DoHBrw 2020	[35]	0.9766	0.99998	0.99993	**0.99998**
DoHBrw 2020	[36]	0.9999	0.99998	0.99993	**0.99998**
NBaIoT2018 Mirai	[37]	0.928	0.99993	0.99993	**0.99995**
NBaIoT2018 Mirai	[38]	0.9998	0.99993	0.99993	**0.99995**
NBaIoT2018 Mirai	[39]	0.99988	0.99993	0.99993	**0.99995**
NBaIoT2018 Mirai	[40]	0.9998	0.99993	0.99993	**0.99995**
UNSW NB15	[33]	0.9819	0.99633	0.98705	**0.99641**
UNSW NB15	[41]	0.9021	0.99633	0.98705	**0.99641**
UNSW NB15	[42]	0.9875	0.99633	0.98705	**0.99641**
UNSW2018 BoTIoT	[31]	0.97	0.99996	0.99988	**1**
UNSW2018 BoTIoT	[35]	0.9996	0.99996	0.99988	**1**
UNSW2018 BoTIoT	[32]	0.99912	0.99996	0.99988	**1**

The bold values highlights the best performing model's accuracy on a particular dataset. If more than one values are bold for a dataset, more than one model achieves the highest accuracy

the NBaIoT2018 Mirai dataset for FDM. The average reduction was 97.03% for both FDM & DFDM and 91.8% average for HAM. The overall dimensionality reduction we achieved for VMFCVD was 95.28

4.3.2 Performance Comparison of VMFCVD with ML Models

The accuracy of VMFCVD's FDM outperformed the ML algorithms used in this work. The VMFCVD's performance, shown in Table 9, shows that VMFCVD outperforms ML algorithms on all the datasets experimented on within this work. However, on eight occasions, ML models reached the accuracy of VMFCVD.

Table 10 shows the accuracy of detecting malicious packets when experimenting with the cluster of FDM. In this performance comparison, DFDM dominates ML models most of the time.

The accuracy of VMFCVD's HAM outperformed the ML algorithms used in this work. The VMFCVD's performance in Table 11 shows that it outperformed ML algorithms on all the datasets experimented on within this work. However, seven out of 110 comparisons, one of the ML models reached the performance of VMFCVD.

4.3.3 Performance Comparison of VMFCVD with State-of-the-art Baselines

In this section, we compare the accuracy of all three modes of VMFCVD with state-of-the-art baselines. Table 12 shows that VMFCVD outperformed other studies in terms of classification accuracy. HAM topped all different modes of VMFCVD and recent ML models, as shown in Table 12. Despite the significant dimensionality reduction in FDM, the performance of FDM is better in most of the recent studies. Although DFDM is dedicated to detecting malicious pack-

ets efficiently, its overall accuracy is comparable to recent studies. It was better in most cases, as shown in Table 12.

Figure 12 depicts the performance comparison of VMFCVD over ML models. The accuracy of ML models ranged from 43% to 100%. The accuracy of VMFCVD was always above 98.7%, with an overall average of 99.82%.

5 Conclusion and Future Work

DDoS attacks continue to undermine the availability of online services despite the enormous effort made by researchers and industries to defend them. Existing systems suffer from high processing overhead and are validated on a limited number of datasets. In this work, the extensive work on feature selection and then systematic approach to the formation and selection of best clusters gives a dimensionally reduced high-quality dataset created from noisy datasets for various VMFCVD modes. VMFCVD takes low processing overhead when the server is under attack. Extensive experiments were performed to evaluate its effectiveness. It is evaluated on the CICIDS2017 (BoT & DDoS), CSE-CIC-IDS2018 (BoT & DDoS), CICDDoS2019 (DNS, LDAP, SSDP & SYN), DoHBrw2020, NBaIoT2018 (Mirai), UNSW2018 BoTIoT, and UNSW NB15 datasets. The experimental results show that VMFCVD outperformed other studies in terms of classification accuracy. The extent to which we have reduced the dataset is maximal compared to any of the previous studies. In some cases, VMFCVD reduced the dataset by 98.2%, maintaining an accuracy of 99.99%. As of future work, we plan to create a generic DDoS and botnet dataset that can be used to train the model when implemented on a live server. We plan to include a module to identify and block devices if multiple malicious network traffic come from them.

Funding The authors did not receive support from any organization for the submitted work.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Kolias, C.; Kambourakis, G.; Stavrou, A.; Voas, J.: Ddos in the iot: Mirai and other botnets. Computer **50**(7), 80–84 (2017). https://doi.org/10.1109/MC.2017.201
2. Mielke, C.J.; Chen, H.: Botnets, and the cybercriminal underground. In: 2008 IEEE International Conference on Intelligence and Security Informatics, pp. 206–211. IEEE (2008). https://doi.org/10.1109/ISI.2008.4565058
3. Wang, A.; Chang, W.; Chen, S.; Mohaisen, A.: A data-driven study of DDoS attacks and their dynamics. IEEE Trans. Dependable Secure Comput. **17**(3), 648–661 (2018). https://doi.org/10.1109/TDSC.2018.2808344
4. Jonker, M.; King, A.; Krupp, J.; Rossow, C.; Sperotto, A.; Dainotti, A.: Millions of targets under attack: a macroscopic characterization of the dos ecosystem. In: Proceedings of the 2017 Internet Measurement Conference, pp. 100–113 (2017). https://doi.org/10.1145/3131365.3131383
5. Humayun, M.; Niazi, M.; Jhanjhi, N.; Alshayeb, M.; Mahmood, S.: Cyber security threats and vulnerabilities: a systematic mapping study. Arab. J. Sci. Eng. **45**(4), 3171–3189 (2020). https://doi.org/10.1007/s13369-019-04319-2
6. Warburton, D.: DDoS Attack Trends for 2020, F5Labs (2020). https://www.f5.com/labs/articles/threat-intelligence/ddos-attack-trends-for-2020. Accessed 6 Aug 2021
7. Costa Gondim, J.J.; de Oliveira Albuquerque, R.; Clayton Alves Nascimento, A.; García Villalba, L.J.; Kim, T.-H.: A methodological approach for assessing amplified reflection distributed denial of service on the internet of things. Sensors **16**(11), 1855 (2016). https://doi.org/10.3390/s16111855
8. Aamir, M.; Zaidi, S.M.A.: DDoS attack detection with feature engineering and machine learning: the framework and performance evaluation. Int. J. Inf. Secur. **18**(6), 761–785 (2019). https://doi.org/10.1007/s10207-019-00434-1
9. Doriguzzi-Corin, R.; Millar, S.; Scott-Hayward, S.; Martinez-del-Rincon, J.; Siracusa, D.: Lucid: a practical, lightweight deep learning solution for DDoS attack detection. IEEE Trans. Netw. Serv. Manag. **17**(2), 876–889 (2020). https://doi.org/10.1109/TNSM.2020.2971776
10. Jia, Y.; Zhong, F.; Alrawais, A.; Gong, B.; Cheng, X.: Flowguard: an intelligent edge defense mechanism against IoT DDoS attacks. IEEE Internet Things J. **7**(10), 9552–9562 (2020). https://doi.org/10.1109/JIOT.2020.2993782
11. Injadat, M.; Moubayed, A.; Nassif, A.B.; Shami, A.: Multi-stage optimized machine learning framework for network intrusion detection. IEEE Trans. Netw. Serv. Manag. (2020). https://doi.org/10.1109/TNSM.2020.3014929
12. Priyadarshini, R.; Barik, R.K.: A deep learning based intelligent framework to mitigate DDoS attack in fog environment. J. King Saud Univ. Comput. Inf. Sci. (2019). https://doi.org/10.1016/j.jksuci.2019.04.010
13. Aamir, M.; Zaidi, S.M.A.: Clustering based semi-supervised machine learning for DDoS attack classification. J. King Saud Univ. Comput. Inf. Sci. (2019). https://doi.org/10.1016/j.jksuci.2019.02.003
14. ur Rehman, S.; Khaliq, M.; Imtiaz, S.I.; Rasool, A.; Shafiq, M.; Javed, A.R.; Jalil, Z.; Bashir, A.K.: Diddos: An approach for detection and identification of distributed denial of service (DDoS) cyberattacks using gated recurrent units (GRU). Future Gener. Comput. Syst. **118**, 453–466 (2021). https://doi.org/10.1016/j.future.2021.01.022
15. Popoola, S.I.; Adebisi, B.; Hammoudeh, M.; Gui, G.; Gacanin, H.: Hybrid deep learning for botnet attack detection in the internet-of-things networks. IEEE Internet Things J. **8**(6), 4944–4956 (2020). https://doi.org/10.1109/JIOT.2020.3034156
16. Ravi, N.; Shalinie, S.M.: Learning-driven detection and mitigation of DDoS attack in IoT via SDN-cloud architecture. IEEE Internet Things J. **7**(4), 3559–3570 (2020). https://doi.org/10.1109/JIOT.2020.2973176
17. Gu, Y.; Li, K.; Guo, Z.; Wang, Y.: Semi-supervised k-means DDoS detection method using hybrid feature selection algorithm. IEEE Access **7**, 64351–64365 (2019). https://doi.org/10.1109/ACCESS.2019.2917532
18. Bawany, N.Z.; Shamsi, J.A.; Salah, K.: DDoS attack detection and mitigation using SDN: methods, practices, and solutions. Arab. J.

Sci. Eng. **42**(2), 425–441 (2017). https://doi.org/10.1007/s13369-017-2414-5
19. Idhammad, M.; Afdel, K.; Belouch, M.: Semi-supervised machine learning approach for DDoS detection. Appl. Intell. **48**(10), 3193–3208 (2018). https://doi.org/10.1007/s10489-018-1141-2
20. Singh, D.; Singh, B.: Investigating the impact of data normalization on classification performance. Appl. Soft Comput. **97**, 105524 (2020). https://doi.org/10.1016/j.asoc.2019.105524
21. Kasun, L.L.C.; Yang, Y.; Huang, G.-B.; Zhang, Z.: Dimension reduction with extreme learning machine. IEEE Trans. Image Process. **25**(8), 3906–3918 (2016). https://doi.org/10.1109/TIP.2016.2570569
22. Gao, L.; Wu, W.: Relevance assignation feature selection method based on mutual information for machine learning. Knowl. Based Syst. **209**, 106439 (2020). https://doi.org/10.1016/j.knosys.2020.106439
23. Hall, M.A.: Correlation-based feature selection for machine learning (1999)
24. Osisanwo, F.; Akinsola, J.; Awodele, O.; Hinmikaiye, J.; Olakanmi, O.; Akinjobi, J.: Supervised machine learning algorithms: classification and comparison. Int. J. Comput. Trends Technol. (IJCTT) **48**(3), 128–138 (2017)
25. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. ICISSp **1**, 108–116 (2018)
26. Sharafaldin, I.; Lashkari, A.H.; Hakak, S.; Ghorbani, A.A.: Developing realistic distributed denial of service (DDoS) attack dataset and taxonomy. In: 2019 International Carnahan Conference on Security Technology (ICCST), pp. 1–8. IEEE (2019). https://doi.org/10.1109/CCST.2019.8888419
27. MontazeriShatoori, M.; Davidson, L.; Kaur, G.; Lashkari, A.H.: Detection of DoH tunnels using time-series classification of encrypted traffic. In: 2020 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress, pp. 63–70. IEEE (2020). https://doi.org/10.1109/DASC-PICom-CBDCom-CyberSciTech49142.2020.00026
28. Meidan, Y.; Bohadana, M.; Mathov, Y.; Mirsky, Y.; Shabtai, A.; Breitenbacher, D.; Elovici, Y.: N-baiot—network-based detection of IoT botnet attacks using deep autoencoders. IEEE Pervasive Comput. **17**(3), 12–22 (2018). https://doi.org/10.1109/MPRV.2018.03367731
29. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B.: Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset. Futur. Gener. Comput. Syst. **100**, 779–796 (2019). https://doi.org/10.1016/j.future.2019.05.041
30. Moustafa, N.; Slay, J.: UNSW-NB15: a comprehensive data set for network intrusion detection systems (UNSW-NB15 network data set). In: 2015 Military Communications and Information Systems Conference (MilCIS), pp. 1–6. IEEE (2015). https://doi.org/10.1109/MilCIS.2015.7348942
31. Ferrag, M.A.; Maglaras, L.; Ahmim, A.; Derdour, M.; Janicke, H.: Rdtids: rules and decision tree-based intrusion detection system for internet-of-things networks. Futur. Internet **12**(3), 44 (2020). https://doi.org/10.3390/fi12030044
32. Prasad, M.; Tripathi, S.; Dahal, K.: An efficient feature selection based Bayesian and rough set approach for intrusion detection. Appl. Soft Comput. **87**, 105980 (2020). https://doi.org/10.1016/j.asoc.2019.105980
33. Sarhan, M.; Layeghy, S.; Moustafa, N.; Portmann, M.: Netflow datasets for machine learning-based network intrusion detection systems. arXiv preprint arXiv:2011.09144 (2020)
34. Alamri, H.A.; Thayananthan, V.: Bandwidth control mechanism and extreme gradient boosting algorithm for protecting software-defined networks against DDoS attacks. IEEE Access **8**, 194269–194288 (2020). https://doi.org/10.1109/ACCESS.2020.3033942
35. Liu, Z.; Thapa, N.; Shaver, A.; Roy, K.; Siddula, M.; Yuan, X.; Yu, A.: Using embedded feature selection and CNN for classification on CCD-INID-V1—a new IoT dataset. Sensors **21**(14), 4834 (2021). https://doi.org/10.3390/s21144834
36. Jafar, M.T.; Al-Fawa'reh, M.; Al-Hrahsheh, Z.; Jafar, S.T.: Analysis and investigation of malicious DNS queries using CIRA-CIC-DoHBrW-2020 dataset
37. Palla, T.G.; Tayeb, S.: Intelligent Mirai malware detection for IoT nodes. Electronics **10**(11), 1241 (2021). https://doi.org/10.3390/electronics10111241
38. Karthik, M.G.; Krishnan, M.M.: Hybrid random forest and synthetic minority over sampling technique for detecting internet of things attacks. J. Ambient. Intell. Humaniz. Comput. 1–11 (2021). https://doi.org/10.1007/s12652-021-03082-3
39. Mafarja, M.; Heidari, A.A.; Habib, M.; Faris, H.; Thaher, T.; Aljarah, I.: Augmented whale feature selection for IoT attacks: structure, analysis and applications. Futur. Gener. Comput. Syst. **112**, 18–40 (2020). https://doi.org/10.1016/j.future.2020.05.020
40. Al-Hawawreh, M.; Moustafa, N.; Garg, S.; Hossain, M.S.: Deep learning-enabled threat intelligence scheme in the internet of things networks. IEEE Trans. Netw. Sci. Eng. (2020). https://doi.org/10.1109/TNSE.2020.3032415
41. Yang, Y.; Zheng, K.; Wu, C.; Niu, X.; Yang, Y.: Building an effective intrusion detection system using the modified density peak clustering algorithm and deep belief networks. Appl. Sci. **9**(2), 238 (2019). https://doi.org/10.3390/app9020238
42. Gao, Y.; Wu, H.; Song, B.; Jin, Y.; Luo, X.; Zeng, X.: A distributed network intrusion detection system for distributed denial of service attacks in vehicular ad hoc network. IEEE Access **7**, 154560–154571 (2019). https://doi.org/10.1109/ACCESS.2019.2948382