Computer Communications 145 (2019) 203–222

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Active learning to detect DDoS attack using ranked features

Rup Kumar Deka [a,*], Dhruba Kumar Bhattacharyya [a], Jugal Kumar Kalita [b]

[a] Department of Computer Science and Engineering, Tezpur University, Napaam, Assam, India
[b] Department of Computer Science, College of Engineering and Applied Science, University of Colorado, Boulder, CO, United States

ARTICLE INFO

Keywords:
Feature selection
Big data
Parallel computing
Active learning
Support vector machine(SVM)
DDoS
Machine learning(ML)

ABSTRACT

Network traffic classification to detect DDoS attacks is challenging in the context of high-speed networks. In this paper, we discuss the need for distributed feature selection in intrusion detection systems using parallel computing. This paper presents a parallel cumulative ranker algorithm to rank the attributes of a dataset for cost-effective classification of network traffic. We use MIT-DARPA, CAIDA, ISCX-IDS and TU-DDoS datasets to validate our method. Our feature ranking algorithm on large datasets (50,000-1,000,000 instances) finds best possible features from the above mentioned datasets and gives high accuracy (92%-97%) in a parallel environment, which takes significantly less time (71%-85% lower) than a sequential environment. We also discuss the importance of active learning to select appropriate instances by an expert module in an unsupervised way to train an SVM binary classifier for detection of DDoS attack traffic. Our approach selects small batches of training samples from a dataset to yield classification of network traffic with high accuracy. Our approach on large data provides better accuracy in classification with fewer training samples. A case study looks into the detection of intrusion in power systems.

1. Introduction

Big institutional networks are continuously adding to the types and numbers of connected devices leading to evermore complex networks that need to be maintained 24 × 7, possibly around the world. The network traffic generated by these devices is voluminous, making it very difficult to monitor and perform appropriate analysis to detect abnormal behavior [1]. In 2003, the total amount of data generated on the Internet was about 5 exabytes, which tripled to 14.7 exabytes in 2008. Today, approximately 5 exabytes of data are produced by users every two days [2]. The data generation rate, the speed of data in the network, and the variety and complexity of data are also increasing tremendously. That is why, this is the age of Big Data. There are many challenges in extracting useful information from large amounts of data or to classify such data. In this paper, we focus on classification of network traffic data only.

Network traffic classification can facilitate the handling of different issues in network management. It is essential for network operators to know what is flowing through their networks so that they can react quickly in support of their business. For example, classification of network traffic can be incorporated in an automated intrusion detection system [3,4] as a core component, or can be used to detect specific abnormalities (for example, Denial of Service attacks) or initiate a re-allocation strategy on network resources for preferred customers [5] automatically.

Distributed Denial of Service (DDoS) attacks have posed serious threats to all kinds of businesses and enterprises. It is very hard to defend against DDoS attacks because of changing attack signatures. Distinguishing legitimate traffic from malicious traffic is hard, and to do so in real time is even more complex. Setting up an appropriate filter mechanism manually is often extremely difficult due to the large number of hosts involved in an attack. It is also impossible to set filters for a zero day attack in a pro-active way [6].

Changing trends in Network Traffic: It will be ideal if a classification technique can consistently discriminate DDoS traffic from normal traffic with high accuracy in all possible scenarios. Several difficulties need to be handled because of changing patterns in DDoS traffic, when embedded in normal traffic. Non-standard ports, disguised ports and network address translation (NAT) additionally increase the difficulties during classification. When traffic is tunneled using different protocols, finding accurate analysis of traffic becomes further difficult, and more invasive inspection by the classifier becomes necessary. Encrypted or encoded traffic adds a security layer, but this limits the ability to extract features. Evolving trends in applications give us different advantages, such as better service qualities, improved service policies and more stringent security policies. These applications use multiple channels, e.g., signaling, video streaming, chat, data transfer and voice calls. Most traffic classification techniques cannot keep pace with such changing trends.

* Corresponding author.
E-mail addresses: rup.deka@gmail.com (R.K. Deka), dkb@tezu.ernet.in (D.K. Bhattacharyya), jkalita@uccs.edu (J.K. Kalita).

https://doi.org/10.1016/j.comcom.2019.06.010
Received 2 May 2019; Received in revised form 28 May 2019; Accepted 22 June 2019
Available online 8 July 2019

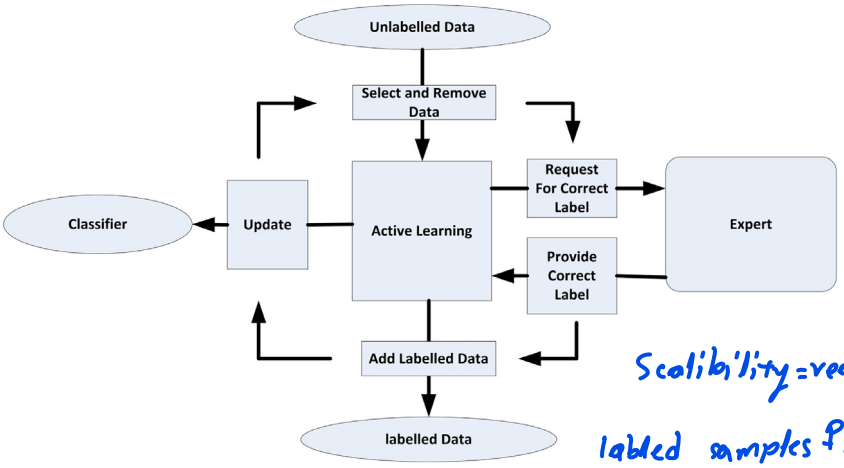


Fig. 1. A Conceptual View of Active Learning.

Scalability Synchronization: The Internet is growing tremendously, including infrastructure and bandwidth capacity of links. In spite of rapid growth, traffic classification needs to work online, providing live information or reacting as per the classification results in real time. Real time traffic classification on such voluminous network traffic requires balancing accuracy, performance and cost. The challenge is to analyze the samples systematically, in a rigorous way to yield high accuracy for all possible scenarios in real (or near real) time. Machine learning techniques require decreasing the latency or limiting the latency of classification during execution. Different methods reduce data by controlling the number of packets in a flow [7,8], or by extracting features. Thus, computational overhead is minimized by reducing the set of features [9] or dimensionality of the dataset. Scalability can be handled with better architectural or framework choices on different available hardware designs. General-purpose graphical processing units (GPGPUs) provide a new paradigm for computation, allowing execution intensive applications to achieve desired performance with minimal hardware costs. As far as software goes, we have to adapt machine learning algorithms such as support vector machines (SVM) to multi-core processing units to yield better scalability.

Classification strategy: The classification results obtained by supervised methods rely on the quality of the labeled samples used for learning. Many supervised and unsupervised methods have been developed for classification of network traffic [10–13]. In supervised methods, proper labeling of samples in real time is usually difficult and expensive since, it requires one to select and label the training samples, as they come. This manual process can slow down the training phase without contributing much to the results. Active learning is an approach to reduce the cost of labeling; the training set is kept as small as possible to avoid redundancy. Active learning is a special case of semi-supervised machine learning. In active learning, the learning process analyzes unlabeled samples to select the most informative patterns and updates the training set repeatedly, depending upon the judgement of a supervisor or expert who assigns labels to the selected unlabeled samples, as shown in Fig. 1. Thus, we can get a small labeled training set at a reduced cost. Existing active learning approaches for classification select the single most informative sample in a single iteration [14,15]. However, in our scenario, the classifier has to be trained for each new labeled sample repeatedly.

1.1. Motivation

In this paper, we address the issue of batch mode active learning to classify network traffic for detection of abnormal traffic, e.g., DDoS attack traffic, where a batch of unlabeled samples is queried at each iteration by considering both uncertainty and diversity criteria [16,17]. Confidence of a supervised algorithm on a considered sample is related to the uncertainty of that sample. Lower the confidence higher the uncertainty. The diversity criterion of an unselected sample measures the distance in the feature space from other selected samples. Using these criteria, we can reduce the redundancy among the selected samples and select only a few labeled samples that can maximize the performance of the classifier [18–21].

In the recent past, many approaches have been developed to identify and block Internet-based attacks. For example, an intrusion detection system (IDS) can be deployed to fight external attacks quite effectively, although there are limitations. Zhang et al. [22,23] enhanced traffic classification performance under the difficult circumstance of having a small sample set of data. To handle a big sample set also, we need to change our approach. However, most current approaches [24–26], and [27] have used the centralized approach for defense.

In our work, we identify a few issues in feature selection in large network datasets in the context of classification of abnormal traffic, especially in the presence of DDoS traffic.

(a) **Feature Ranking:** Ranking the features may obtain an optimal feature subset for classification to reduce high-dimensional data in machine learning problems.
(b) **Scalability:** Handling the scalability issue properly will enable the processing of voluminous data in near real time.
(c) **Distributed Approach:** Going beyond the centralized way, distributed approaches are likely to allow treatment of a large volume of data in real time.

Classification of high speed traffic requires a large number of resources for training. It is often necessary to sample a small portion of the data and train the classifier without any deterioration in performance. Such sampling has been established to be suitable for classification of streaming network traffic. According to Morgan [20], state-of-the-art active learning based classification of streaming network traffic has achieved accuracies in the range of 90% or higher. He also concluded that limiting the labeling cost or labeling only a few samples did not affect performance negatively. Binary SVM can be used to find the uncertainty and diversity in two-class classification problems. In some active learning approaches [28], the authors have presented a technique that selected the most uncertain samples by choosing the samples closest to the current separating hyper-plane. The label of the most interesting/ambiguous unlabeled point was queried at each step.

1.2. Contribution

We contribute the following.

- In this paper, we develop a method to rank the features in a large dataset of network traffic, using a parallel approach on GPUs. After rank the features, we choose the top features as per user requirement for classification of the network traffic to identify DDoS attack traffic. Our approach is cost effective in comparison to several other competing methods. The Proposed frameworks exhibit three important features.

  (1) Unlike a wrapper approach, our approach does not need to use fixed classifier.
  (2) Our approach is cost effective due to the use of a parallel ranking approach.
  (3) Our approach yields high accuracy.

- We present an active learning approach to select appropriate instances of traffic from batches of training data to classify network traffic with high accuracy. We establish the method both theoretically as well as experimentally using several benchmark datasets. Our approach is effective in recent scenarios and demonstrates high accuracy consistently.
- We present a case study on DDoS attack on electrical power systems, which constitute critical infrastructure.

1.3. Organization

The rest of the paper is organized as follows. Section 2 includes discussion on several different existing approaches. Details about feature ranking and active learning are reported in Section 3 and Section 4, respectively. In both sections, our proposed frameworks, procedural designs, and experimental findings on different datasets are elaborated. In Section 5, we provide a case study on detection of DDoS attacks on an electric power system. Section 6 provides the conclusion and future work.

2. Related work

2.1. Feature ranking

A number of methods have been developed to classify abnormal or DDoS traffic from normal network traffic. Chawla et al. [32] discussed a method to discriminate legitimate traffic generated by a flash crowd from DDoS traffic using Pearson's product moment correlation coefficient (PPMCC). However, the method did not focus on choosing features necessary for effective classification. Zhang et al. [23] developed a non-parametric enhanced classification scheme using correlation computation. A drawback of this approach was that it had access to only a few training samples.

Luo and Jingbo [24] developed an intrusion detection method by creating a four-angle star based visualization technique to generate features. In this method, intrusion detection was treated as a 5-class classification problem and evaluated using the KDDcup99 dataset, considering a small number of features.

Bravi et al. [34] associated scores to features to assess their relevance. They used regression assuming that the process underlying the generated data can be approximated by a continuous function. Their approach solved a minimum zero-norm inversion problem using a neural network. They performed concave approximation of the zero-norm function and defined a smooth, global optimization problem to be solved, to estimate the relevance of the features.

Ravale et al. [31] combined K-means clustering and RBF kernel Support Vector Machine (SVM) in a classification module to decrease the number of attributes. They tested this approach on the KDDcup99 dataset [43]. Cao et al. [40] used an adaptive SVM with Principal Component Analysis (PCA) to find the relevance of the features. Yusof et al. [33] used a hybrid KNN-SVM method for classification to detect and predict DDoS traffic.

Jia et al. [42] proposed and established a DDoS attack detection method with hybrid heterogeneous multiclassifier ensemble learning. The algorithm used is a heuristic detection algorithm based on Singular Value Decomposition (SVD) [44,45], validated using the KDDcup99 dataset.

In Table 1, we compare a few approaches for DDoS traffic detection. We observe the following.

- Faster and appropriate ranking of each feature for a given class can enhance the classification performance significantly.
- Statistical techniques such as correlation and dispersion measurement can help estimate the rank of a feature for a given class effectively.
- Instead of testing a detection method on a small amount of data, we need to look for a bigger dataset for better training.
- We need to use datasets that reflect current networks, so that the approach can be extended to handle Big Data now.
- In the context of high speed network traffic, we need to achieve cost effective classification. So, feature selection using parallel computation is likely to be helpful.

2.2. Active learning

In the past years, a substantial amount of work has been carried out for improved Internet traffic classification [46]. For traffic policing, network security and traffic management, IP (internet protocol) traffic classification is essential. Some widely used techniques are port-based traffic classification, payload based traffic classification (Deep Packet Inspection), statistical approaches and machine learning approaches [47–52].

Port-based traffic classification is a fast and simple method, but several studies have shown that it does not perform consistently [53]. Payload-based classification inspects the contents of each packet to identify the application. It uses unique payload signatures for further detection and classification with high accuracy. DPI (deep packet inspection) adds complexity to the solution of security, i.e., it uses firewalls, intrusion detection systems, session border controllers, and honeypots/nets arranged at network margins to defend security boundaries. These processes require regular monitoring, configuration changes, and log analysis [54–56]. The effectiveness of such 'deep packet inspection' techniques is diminishing because such packet inspection relies on two related assumptions.

- Third parties unaffiliated with either source or recipient are able to inspect each IP packet's payload.
- The classifier knows the syntax of each application's packets.

Usually, customers may use encryption to obfuscate packet contents (including TCP or UDP port numbers), and governments may impose privacy regulations to constrain the ability of third parties to lawfully inspect payloads. Thus, the first assumption may not hold. The second assumption makes it necessary that commercial devices will need to be updated frequently to stay ahead of changes in application packet payload formats.

Researchers are looking for new developments in traffic classification. One observation is that at the network layer, Internet traffic exhibits statistical properties considering characteristics such as the distribution of flow duration, flow idle time, packet inter-arrival time and packet lengths. These are unique for certain classes of applications and enable different source applications to be distinguished from each other. The relationship between the class of traffic and its observed statistical properties has been noted in [50]. The authors analyzed and constructed empirical models of connection characteristics such as bytes, duration, and arrival periodicity for a number of specific TCP applications. Dewes et al. [51] analyzed Internet chat systems by focusing on the characteristics of the traffic in terms of flow duration, packet inter-arrival time, packet size and byte profile.

Table 1
Comparison of Some Existing Works.

Author and Year	Objective	Approach used	Technique used	Tool/Dataset used
Zhang et al. [23], 2013	Traffic classification	Non-parametric	Correlate Information	17-filter[a] and Tstat[b] tools, sigcomm traces[c], IBNL traces [29], keio trace, wide trace[d], and IP trace dataset [30]
Luo and Jingbo [24], 2014	Intrusion detection with fewer features	Visualization of features	Compute distance between samples	KDDcup99[e]
Ravale et al. [31], 2015	Intrusion detection	Reduction of number of attributes	K Means clustering and RBF kernel Support Vector Machine	KDDcup99
Chawla et al. [32], 2016	Discrimination of DDoS attacks from flash events	Correlation	Pearson's Product Moment	1998 FIFA World Cup[f], CAIDA[g], MIT DARPA[h] and GENI [32]
Yusof et al. [33], 2016	Detection and prediction of DDoS attack	Hybrid classification	KNN-SVM	KDDcup99
Bravi et al. [34], 2017	Feature ranking	Feature scoring	Concave approximation	Poland electricity load [35], [36], diabetes [37], Santa Fe laser [38], [39], housing[i], abalone dataset[i], cpusmall[i].
Cao et al. [40], 2017	Traffic classification	Machine learning	SVM and PCA	Andrew Moore [41]
Jia et al. [42], 2017	DDoS attack detection	Ensemble learning	Hybrid heterogeneous multiclassifier and Singular Value Decomposition (SVD)	KDDcup99
Our Work	Ranking of network traffic features	Parallel computation	Correlation, Dispersion	MIT-DARPA, CAIDA, ISCX[j] and TU-CANNON tool[k]

[a] http://l7-filter.sourceforge.net.
[b] http://tstat.tlc.polito.it.
[c] http://www.cs.umd.edu/projects/wifidelity/tracing.
[d] http://mawi.wide.ad.jp/mawi/.
[e] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.
[f] http://ita.ee.lbl.gov/html/contrib/WorldCup.html.
[g] https://www.caida.org/data/.
[h] https://www.ll.mit.edu/ideval/data/.
[i] http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets.
[j] http://www.unb.ca/cic/research/datasets/ids.html.
[k] http://agnigarh.tezu.ernet.in/~dkb/tucannon/index.html.

Lang et al. [57,58] also observed distinctive traffic characteristics such as the distributions of packet lengths and packet inter-arrival times, for a number of Internet applications. The results of these works have stimulated the development of new classification techniques based on statistical properties of traffic flow. The need to deal with traffic patterns, large datasets and multi-dimensional spaces of flow and packet attributes is one of the major reasons for the introduction of machine learning techniques in this field.

Several other supervised methods have been proposed for classification of network traffic [23,59,60]. The classification results obtained by these methods relied on the quality of the labeled samples used for learning. Supervised classifiers provide better classification, if there are accurate and adequate amounts of labeled samples for training. Park et al. [61] showed that accuracy was sensitive to site-dependent training datasets, while Erman et al. [62] showed different accuracy results between the two data traces analyzed with the same ML algorithms. Active learning is a semi-supervised approach which can address these issues. However, the amount of work on active learning based classification of network traffic has been limited till this time.

3. Feature selection and ranking using parallel computation

Being able to classify network traffic correctly and quickly has great impact on areas such as performance monitoring, QoS, user behavior analysis, user accounting and intrusion detection [63–65]. Effective and efficient classification algorithms that can work with large amounts of data are also useful in areas such as microarray analysis [66,67], image classification [68,69], face recognition [70,71] and text classification [72,73]. Many datasets that need to be processed also are high dimensional, i.e., they have a large number of features. Feature selection often leads to better classification of data by enabling better extraction of relevant patterns.

Generally, in a single dataset, feature selection is performed in a centralized setting. However, data can naturally occur in a distributed manner or we can partition and store the data for handling in a distributed manner. Thus, an approach to feature selection should be able to handle voluminous data or multiple partitions of a dataset concurrently. There are several ways to distribute the feature selection task [74]. According to Bolón-Canedo et al. [2], there are four reasons to apply a distributed feature selection procedure.

- The data may be in a single large set.
- The data may be in different sets in different locations.
- In real time, infinitely large volumes of data may be arriving continuously.
- The dataset may not be particularly large, but different feature selection methods can be applied in parallel to evaluate what works well, and whether it helps to use different sets of features in an ensemble situation.

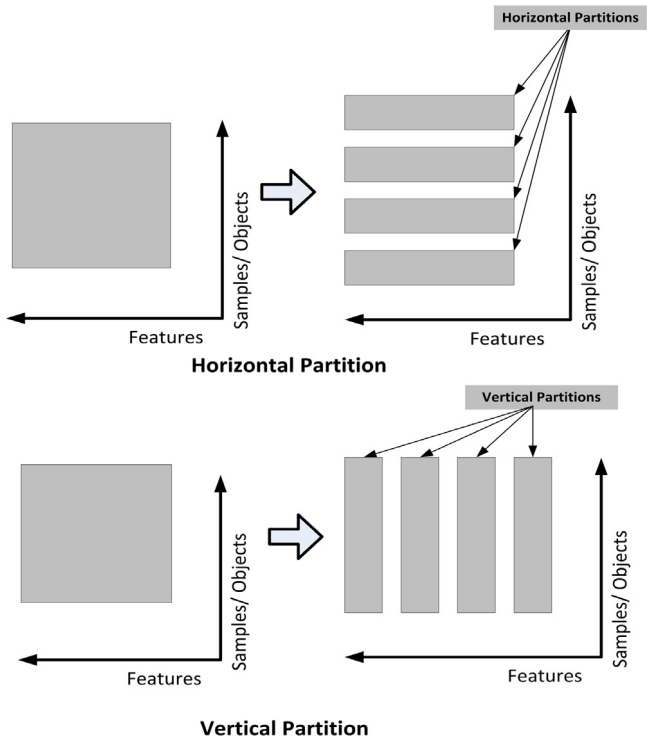


Fig. 2. Two ways of partitioning a dataset.

3.1. Feature selection

Research on feature selection was initiated around 1960 [75]. Feature selection can be defined as a technique to choose a subset of relevant features by removing irrelevant and redundant features [76] from a dataset to build learning models [77]. Thus, it includes a search process to find a best feature subset from all possible subsets. An evaluation measure is applied to estimate and evaluate the relevance of each feature subset for a given class. We can select or rank features in three ways: wrapper, filter, and embedded methods.

3.2. Feature selection in intrusion detection

A number of methods for network traffic classification have been proposed [46,78,79]. Traffic classification is normally an essential component in QoS control [80] and intrusion detection [81,82]. With the popularity of cloud computing [83,84], the number of applications deployed on the Internet is quickly increasing. The types of techniques applied for classification for intrusion defense most commonly include decision trees, artificial neural networks (ANN), naive Bayes and fuzzy set theoretic approaches. These approaches can be further categorized based on prior knowledge used during feature selection, into three, viz., supervised, semi-supervised and unsupervised.

DDoS attacks represent a major type of intrusion. The size and speed of DDoS attacks are increasing day by day [85]. The perpetrators of this type of attack are always changing their style of attack to consume more bandwidth. There are two major issues that hamper network data analysis for DDoS attack detection: (i) A huge amount of data can be loaded in the buffer within a short time frame, and (ii) Because of changes in the pattern of traffic, data become heterogeneous, making it harder to mine useful information or pattern.

An intrusion detection system needs to achieve high accuracy with efficient computation. Feature selection can provide an edge in intrusion detection in achieving maximum performance. We select a subset of relevant features to build the defense system. Detecting attacks in real time is often difficult due to the huge amount of data flowing on the Internet. Feature selection can reduce the amount of computation and model complexity. It can help select a minimum number of features from the original feature set to achieve high detection accuracy [86]. An effective feature subset can improve the training and testing times of an intrusion detection system and can make the system lightweight, and hence suitable to perform in real time.

3.3. Feature selection in parallel computing

In recent years, many efforts have been made to exploit GPUs (Graphic Processing Units) as a parallel computing environment [87,88]. Morán-Fernández et al. [89] compared centralized and distributed feature selection methods. According to them, by partitioning the dataset vertically or horizontally we can achieve higher classification performance at significantly reduced running time. Fig. 2 shows these two partitioning approaches. Ramírez-Gallego et al. [90] designed the Fast Minimum Redundancy Maximum Relevance (Fast-mRMR) algorithm and implemented it on several different platforms, namely, CPU for sequential execution, GPU for parallel computing, and Apache Spark for distributed computing using Big Data technologies.

We can distribute the computational load of a voluminous network traffic dataset among the cores of a GPU and apply feature selection approaches to each core locally. The features selected can be put together and ranked on a cumulative basis. This distributed parallel approach can handle a large dataset and process it in near real time without compromising quality.

3.4. Parallel Cumulative Rank (PCR): The proposed method

We aim to develop a cost effective method to classify large volume network traffic for detection of DDoS attacks by operating over an optimal subset of features in a parallel computing environment.

3.5. Framework

Fig. 3 shows our proposed framework as a prototype model. The whole process can be visualized in terms of three major tasks, as discussed below.

3.5.1. Feature ranking

This work starts by ranking the features for each partition of a dataset. The rank of a feature defines the relevancy and non-redundancy of an attribute or feature. Later, a global or cumulative ranking is computed by combining the individual ranks for each partition. Higher the rank, the higher the likelihood of being included in the feature subset for classification. There are three sub-tasks, performed in parallel. The sub-tasks are described next.

(a) Pre-processing. To build the prototype, we choose five attributes of network traffic, source IP address (f1), destination IP address (f2), source port number (f3), destination port number (f4) and frame length (f5) of each traffic sample. Using *editcap*[1] and *Tshark*,[2] we obtain the values of these attributes from traffic data containing about 1,000,000 packets. An empirical study was conducted to observe the behavior of our model by varying the number of partitions. Our observation is that the proposed framework is scalable for any number of attributes or features. We check the accuracy behavior as we vary the partition numbers for each dataset.

(b) Dataset Partitioning. To perform distributed parallel computation, we partition the network traffic data vertically. We vary the number of partitions and compute the rank of each feature in each partition. We also track execution time in both a sequential environment and in a parallel environment.

[1] https://www.wireshark.org/docs/man-pages editcap.html.
[2] http://www.wireshark.org/docs/man-pages/tshark.html.

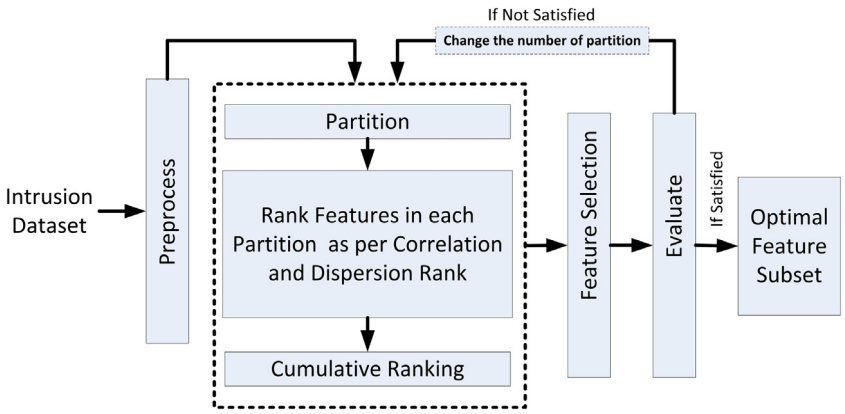


Fig. 3. Proposed Framework of PCR.

(c) Parallel Ranking. Each partition is processed on a machine with GPUs without any overlap. The rank calculation is performed for each partition individually. We consider both relevance and redundancy while computing the rank of an attribute. Two methods are used to derive the rank of each feature.

Definition 1: Correlation rank R_i is defined as the rank of a feature f_i, given by the correlation coefficient Cor_i over all other features of the feature subset S' for the given class C_i. This coefficient gives a measure of the correlation of the feature f_i with all other features from the same subset.

Definition 2: Dispersion rank D_i is defined as the rank of a feature f_i, given by the dispersion coefficient or index of dispersion $Disp_i$ over all other features of the feature subset S' for the given class C_i. This coefficient gives a measure of how the instances of the feature f_i are clustered (homogeneous) or dispersed.

Definition 3: Cumulative rank Cum_i is defined as the global rank of a feature f_i over all other features of the feature subset S' for the given class C_i. This rank is the cumulative rank considering both correlation rank and dispersion rank of the feature f_i of the feature subset S' for the given class C_i.

c.1 *Correlation rank:* We use a correlation measure to compute dependency of an attribute on all other attributes. The rank of an attribute can be computed based on the correlation coefficient of a given class. The Pearson correlation coefficient p is a measure of the linear correlation between two variables x and y. The Pearson correlation coefficient can take values from −1 to +1. A value of +1 shows that the variables are perfectly linearly related by an increasing relationship, a value of −1 shows that the variables are perfectly linearly related by decreasing relationship, and a value of 0 shows that the variables are not linearly related to each other.

$$p = \frac{(n\sum(x*y) - (\sum x)*(\sum y))}{\sqrt{(n\sum(x^2) - (\sum x)^2)*(n\sum(y^2) - (\sum y)^2)}} \quad (1)$$

For a set of m features, we can compute m-1 number of p-values for each of the features to find the relevancy for that class. The mean p-value for one feature gives the relevancy of that feature w.r.t. all other features. Thus, a rank among all the features can be established.

c.2 *Dispersion rank:* Dispersion rank can be defined as a measure used to quantify whether a set of observed instances of an attribute are clustered or dispersed for a given class. A dispersion measure gives us a score of spread in the values of an attribute in a dataset with respect to the mean value of that attribute. In other words, it is a measure used to evaluate whether a set of observed occurrences are clustered tightly or dispersed. A higher dispersion value means a bigger spread. This concept can be applied to rank an attribute, so that we can get more relevant and non-redundant features. The dispersion value q of a feature is given by

$$q = \frac{variance}{mean} = \frac{\sigma^2}{\mu} \quad (2)$$

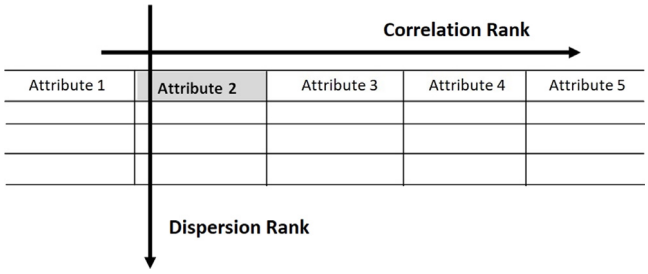


Fig. 4. Abstract Idea of Rank Evaluation.

The idea is to find a rank for the feature depending on the uniqueness of an attribute in a collection of samples. If any other measure can provide similar rank, we may be able to substitute one for the other. In Fig. 4, we see that correlation rank provides us the horizontal uniqueness among the attributes and dispersion rank provides vertical uniqueness of the attribute.

Cumulative rank. Cumulative rank is defined as the rank we get after combining both correlation and dispersion ranks to compute the relevancy better for an attribute for a given class, considering all partitions. The correlation rank provides us the horizontal relevancy and the dispersion rank provides vertical relevancy of the attribute for that class. The mean value of both ranks for a single attribute provides one rank over all attributes for a single partition of the dataset. So, the ranks of all the partitions are accumulated to produce a global rank. With variations in the numbers of vertical partitions, correlation rank and dispersion rank also vary, and so does the cumulative rank. It is important to note that the rank of a set of features for a fixed dataset does not remain the same if we change the number of partitions, and when we do so, the classification accuracy also varies. For a fixed dataset, we may need to tune the number of vertical partitions to obtain better classification accuracy, and parallel computation helps such tuning easily.

Consider the example shown in Fig. 4,

- Let the Correlation Rank of Attribute 2 in the first partition be 3, and the Dispersion Rank of Attribute 2 in the first partition be 4.
- Then the rank of Attribute 2 in the first partition ($\lceil((3+5)/2)\rceil$) = 4

Source Address (f1)	Destination Address (f2)	Source PORT (f3)	Desitnation PORT (f4)	Frame Length (f5)
192.[redacted].122	192.[redacted].108	22	1220	60
192.[redacted].108	192.[redacted].122	1220	22	102
192.[redacted].122	192.[redacted].108	22	1220	60
192.[redacted].122	192.[redacted].108	22	1220	102
192.[redacted].122	192.[redacted].108	22	1220	60
192.[redacted].122	192.[redacted].108	22	1220	102
192.[redacted].108	192.[redacted].122	1220	22	126
192.[redacted].118	203.[redacted].75	4509	80	60
203.[redacted].75	192.[redacted].118	80	4509	867
192.[redacted].118	203.[redacted].75	4509	80	60
192.[redacted].118	203.[redacted].75	4509	80	60
192.[redacted].118	203.[redacted].75	4510	80	62
192.[redacted].122	192.[redacted].108	22	1220	60

Fig. 5. A Snapshot of the Dataset.

- Suppose there are a total of 5 partitions. The rank values from all partitions of Attribute 2 are 4, 3, 4, 1, and 5.
- So, the Cumulative Rank is $(\lceil((4+3+4+1+5)/5)\rceil) = 4$.

3.5.2. Feature selection

The feature ranking process provides us a rank for each feature. We select a subset of best features using a cut-off point in the rank, which can be chosen heuristically supported by empirical results. We choose three best attributes for classification with respect to the cutoff threshold.

3.5.3. Classification

To evaluate the performance of our ranking approach, we choose a set of five well-known classification algorithms for evaluation. We also check how the accuracy changes as we vary the number of vertical partitions. We check the accuracy of the feature ranking process using five well-known algorithms: Fine KNN (A) [91], Linear Discriminant (B) [92], Logistic Regression (C) [93], Boosting (D) [94], and Complex Decision Tree (E) [95]. Taking one partition with around 100,000 packets from each dataset, we apply the feature ranking process to get the top three features and use these features for classification. We choose five classification algorithms to obtain the accuracy of our proposed ranking algorithm on any dataset in the given framework. Each of these classification algorithm's works differently, and they may give different accuracies on the same dataset. We do not focus on the working principles of the classification algorithms. These classification algorithms can vary depending on availability.

The following propositions are true for Parallel Cumulative Rank (PCR).

Proposition 1: A subset of features S' with high correlation rank for a given class C_i is relevant to the class, i.e., $S' \odot C_i$ where $\odot$ represents relevance (dependence) relationship due to high correlation.

Explanation: Correlation rank is used to assess the significance of the relation between two or more variables. Suppose $f_1, f_2, \ldots, f_n$ are features or attributes in the subset of features S' of the given class C_i. If the Pearson correlation coefficient is close to $+1$ or -1 for f_1 for the given class C_i [as Pearson correlation coefficient value ranges from $+1$ to -1], we say that f_1 has relevance to all other features of that subset of features for the same class [96,97]. Features with higher correlation rank have higher relevancy.

Proposition 2: A subset of features S' with low dispersion rank for a given class C_i is relevant to the class, i.e., $S' \odot C_i$, where $\odot$ represents relevance relationship due to low dispersion.

Explanation: Dispersion coefficient (index of dispersion) is a measure to evaluate how the values or instances of an attribute are clustered or dispersed for a given class. Suppose f_1 is a feature or attribute in the subset of features S' of the given class C_i. The index of dispersion is used to test for homogeneity of the observations of an attribute [98]. When the coefficient of dispersion is low, the instances of feature f_1 of class C_i are said to be "under-dispersed (more homogeneous)" or otherwise, they are "over-dispersed (less homogeneous)". The lower the coefficient value the higher is the dispersion rank for the given class C_i. A more highly homogeneous attribute would be more relevant for the same class.

Proposition 3: A subset of features S' with high cumulative rank for a given class C_i is relevant.

Explanation: Suppose f_1 is a feature or attribute in the subset of features S' of the given class C_i. If it has higher correlation rank and also higher dispersion rank, the cumulative rank or relevancy is higher for class C_i than all other attributes from the same subset of features, S'.

3.6. Experimental set-up and results

We use the following datasets and environment to carry out experiment for the evaluation of PCR.

(a) *Datasets:* We use four datasets viz., MIT-DARPA,[3] CAIDA-2007,[4] ISCX,[5] and TUDDoS.[6] Using *editcap* and *Tshark*, we pre-process the three groups of packets with 50,000, 100,000, and 1,000,000 packets from each dataset. In each group, we combine 60% normal traffic and 40% DDoS attack traffic. The source IP addresses and destination IP addresses have been converted into decimal values. A snapshot of the dataset is shown in Fig. 5.

(b) *Workstation Configuration:* We perform our experiments on a workstation with a 2.30 GHz processor, 64 GB RAM and a 64 bit Windows 10 operating system. For parallel computing, we use

[3] https://www.ll.mit.edu/ideval/data/.
[4] https://www.caida.org/data/.
[5] http://www.unb.ca/cic/research/datasets/ids.html.
[6] http://agnigarh.tezu.ernet.in/~dkb/tucannon/index.html.

Table 2
Execution time for feature ranking for different conditions.

Experiment	Partition number	Environment	Worker number	Average execution time (in seconds)		
				50,000 packets	100,000 packets	1,000,000 packets
1	1	Sequential	0	67.76	73.84	82.30
2	8	Sequential	0	68.45	75.57	83.43
3	1	Parallel	5	10.46	10.85	23.45
4	2	Parallel	5	10.46	10.87	23.48
5	4	Parallel	5	10.51	10.91	23.49
6	6	Parallel	5	10.52	10.93	23.52
7	8	Parallel	5	10.55	10.94	23.54
8	10	Parallel	5	10.57	10.97	23.58

NVIDIA Quadro K620 GPU with a total available graphic memory of 34,778 MB, dedicated video memory of 2,048 MB and 384 cores.

(c) *Platform Used:* We use MATLAB R2016a 64 bit edition to perform our experiments. This version of MATLAB has a parallel computing toolbox. We have used 10 workers in the work environment, to run our evaluation process concurrently. These workers are MATLAB computational engines that run locally. The whole experiment is performed with workers present in a single machine in a distributed model using the parallel computing toolbox. We assume that the communication time required to send the traffic data to a single worker node of the GPU from the CPU is constant.

3.6.1. Results

Initially, we study the effect of partitioning the dataset and working in parallel on the execution time of the feature ranking procedure. In Table 2, we see that if we vertically partition the dataset, the execution time is increased for sequential computing and it gets reduced significantly when we do it in a parallel environment. It is important to see how execution time changes if we change the number of workers for a fixed number of partitions.

In Table 2, we observe the following.

T_{seq} (execution time in sequential environment) for 50,000 packets with 8 partitions with 5 workers is 68.45 s
T_{par} (execution time in parallel environment) for 50,000 packets with 8 partitions with 5 workers is 10.55 s

$$\begin{aligned}&\text{Reduction in time (50,000 packets)}\\&= ((T_{seq} - T_{par})/T_{seq}) * 100\\&= ((68.45 - 10.55/68.45) * 100)\\&= 84.58\%\end{aligned}$$

Similarly,
Reduction in time (100,000 packets) = 85.52%
Reduction in time (1,000,000 packets) = 71.78%

In Table 3 and Fig. 6, we see the changes in time as we vary the number of workers. It takes less time when the number of workers is high. Table 4 and Fig. 7 show that parallel computation time does not vary much with variation in the number of partitions. We also see the change in execution time needed for ranking of the features by changing the number of partitions for a fixed number of workers and hence the speed-up, which increases a bit. This study gives us a direction for the remaining experiments.

From Table 2 onward, experiments are executed on 100,000 packets of network traffic. For classification, we keep the number of workers fixed and change the number of partitions to see the change in rank along with accuracy. We do not increase partition size. However, if we increase the partition size, it would not affect the execution time (Fig. 6). But if we increase the number of workers, it reduces the execution time (Fig. 5). The accumulation of the feature rank from each partition is just a data gathering process in parallel computing. Whatever the overhead it causes, it is small and can be ignored due to the advantages gained from parallel computing.

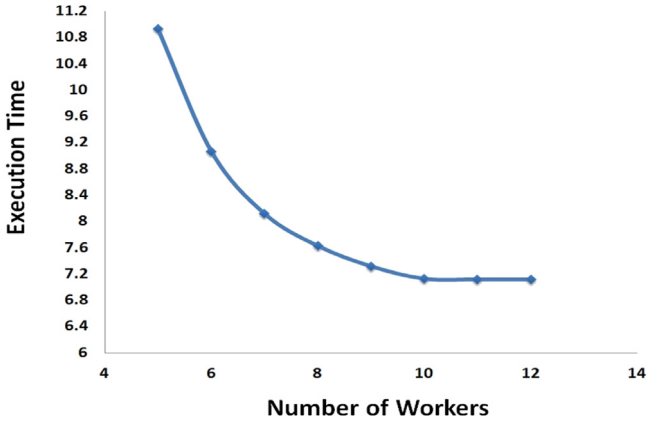


Fig. 6. A graph for execution time vs. number of workers.

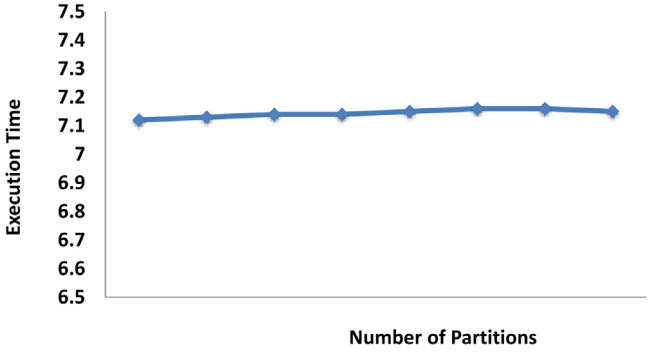


Fig. 7. A graph for execution time vs. number of partitions.

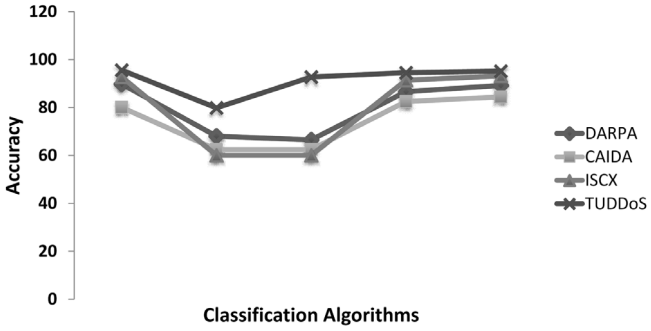


Fig. 8. Accuracy for different algorithms with one partition (whole) for all datasets.

Table 5 and Fig. 8 show the accuracies. Accuracy results are given in Table 6 to Table 9, and Fig. 9 to Fig. 12. In these tables, we see that for different numbers of partitions, the same accuracy level is given by the classifiers. It is because the rank of features is the same. Suppose,

Table 3
Execution time variation due to variation in the number of workers for fixed partitions.

Number of partitions = 6								
Workers	5	6	7	8	9	10	11	12
Avg. Exec. Time (sec)	10.93	9.07	8.12	7.63	7.32	7.13	7.12	7.12

Table 4
Execution time for feature ranking due to variation in number of partitions for fixed workers.

Number of worker = 10								
Partitions	5	6	7	8	9	10	11	12
Avg. Exec. Time (sec)	7.12	7.13	7.14	7.14	7.15	7.16	7.16	7.15

Table 5
Classification accuracies for one partition for four datasets.

Datasets	Feature ranking					Classification accuracy (For Three Top Features)				
	f1	f2	f3	f4	f5	A	B	C	D	E
MIT-DARPA	4	1	5	2	3	89.8	68.0	66.5	86.6	89.2
CAIDA	1	2	4	5	3	80.0	62.4	62.4	82.5	84.4
ISCX	2	1	4	3	5	92.8	60.0	60.0	91.4	93.2
TU DDoS	2	3	1	4	5	95.6	79.8	92.7	94.5	95.2

Table 6
Classification accuracies for different partitions for MIT-DARPA dataset.

Number of partitions	Feature ranking					Classification accuracy (For Three Top Features)				
	f1	f2	f3	f4	f5	A	B	C	D	E
6	2	1	4	5	3	80	62.4	62.4	84.6	82.8
7	4	2	5	2	3	89.8	68.0	66.5	86.6	88.9
8	5	3	5	1	2					
9	5	1	5	5	5	94.8	67.1	67.2	87.4	91.6
10	**5**	**4**	**2**	**1**	**3**	**96.6**	**76.0**	**71.7**	**88.0**	**92.0**

Table 7
Classification accuracies for different partitions for CAIDA dataset.

Number of partitions	Feature ranking					Classification accuracy (For Three Top Features)				
	f1	f2	f3	f4	f5	A	B	C	D	E
6	1	2	5	5	3	93.5	71.8	79.1	60.0	94.7
7	2	1	4	5	4	94.5	67.9	68.6	60.0	95.3
8	2	4	5	3	2	93.7	66.1	69.5	92.7	95.1
9	**1**	**5**	**2**	**4**	**4**	**97.8**	**84.4**	**84.9**	**98.2**	**97.4**
10	3	4	5	2	1	93.7	66.1	69.5	92.7	95.1

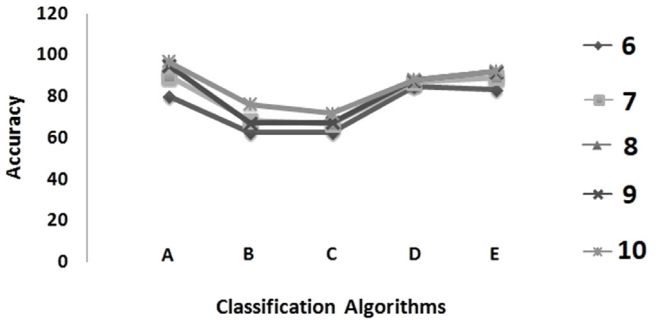


Fig. 9. Accuracy for different algorithms vs. number of partitions for MIT-DARPA dataset.

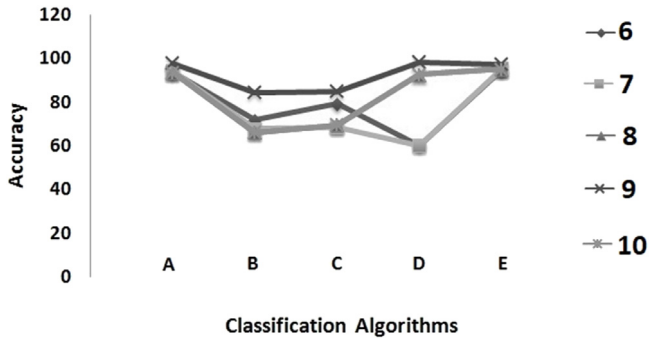


Fig. 10. Accuracy for different algorithms vs. number of partitions for CAIDA dataset.

we get ranks for the five attributes as 4, 2, 5, 2, 3. Then we select the second, fourth and fifth attributes for classification. Again suppose, we get ranks for the five attributes as 5, 1, 5, 5, 5. Then we select first, second and third as the top three attributes. We see for each dataset that the accuracy is the highest for one value of the number of partitions. The number of partitions can be different for different datasets to get high accuracy. The classification accuracy we get after partitioning the datasets is higher than without partitioning (see Figs. 10 and 11).

In Table 6, algorithm E seems to show very low variation in accuracy, which is likely due to the nature of that algorithm. The discussion of these classification algorithms is out of the scope of this article. In Table 8, Algorithm C is 25% less accurate than Algorithm A. There is a variation in accuracy along with other algorithms due to change in the number of partition (see Table 7).

Discussion. Based on the experiments already discussed in this paper, we enumerate a few of our observations.

Table 8
Classification accuracies for different partitions for ISCX dataset.

Number of partitions	Feature ranking					Classification Accuracy (For Three Top Features)				
	f1	f2	f3	f4	f5	A	B	C	D	E
6	1	2	4	3	5	95.7	72.0	77.6	60.0	96.1
7	1	3	4	2	5					
8	3	1	2	5	4	96.4	67.8	68.7	60.0	96.3
9	**1**	**4**	**2**	**3**	**5**	**97.2**	**84.4**	**84.9**	**97.6**	**96.2**
10	1	3	2	5	5	96.4	67.8	68.7	60.0	96.4

Table 9
Classification accuracies for different partitions for TUDDoS dataset.

Number of partitions	Feature ranking					Classification accuracy (For Three Top Features)				
	f1	f2	f3	f4	f5	A	B	C	D	E
6	3	1	4	5	2	94.3	89.9	95.7	94.2	93.5
7	3	1	4	4	5	95.4	79.8	96.7	95.4	94.0
8	2	1	3	4	5					
9	**4**	**2**	**3**	**1**	**5**	**96.2**	**90.8**	**97.5**	**98.2**	**96.8**
10	1	3	4	2	5	95.0	80.4	91.4	93.6	94.9

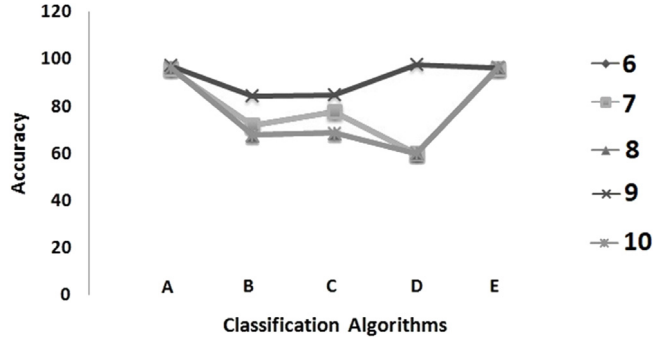


Fig. 11. Accuracy for different algorithms vs. number of partitions for ISCX dataset.

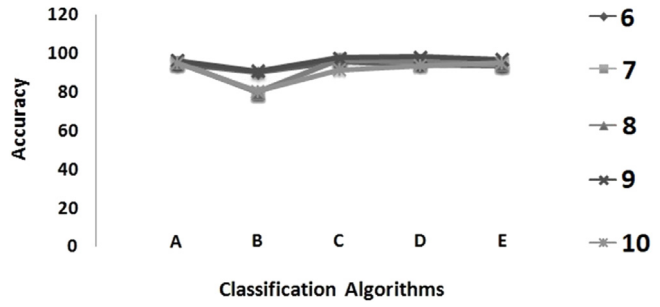


Fig. 12. Accuracy for different algorithms vs. number of partitions for TUDDoS dataset.

- Classification accuracy depends on the selection of an appropriate subset of features.
- An appropriate partitioning of the dataset can help find the best possible feature subset.
- An appropriate selection of the number of worker nodes can help achieve better speed-up.
- An inappropriate partitioning may lead to poor speed-up.
- The selection of the number of worker nodes does not influence classification accuracy.
- An appropriate dataset partitioning and worker node selection can help significantly in the selection of the best possible feature subset, and hence achieve better classification accuracy.
- We experiment using 6–10 partitions of a single dataset. After a ranking algorithm is applied, a specific partition gives us the best feature subset. We select this best subset. Considering the execution time vs. the number of partitions graphs (Fig. 7 and Table 4), we see that the best execution time for our system occurs with 10 partitions for a dataset.
- We execute our ranking algorithm a number of times to decide on the appropriate partition number that gives the best feature subset. This process seems to add significant overhead to our work. However, in a parallel distributed environment, this overhead is expected to be negligible. In our experiment, the execution time for the ranking algorithm for 100,000 packets with 6 partitions and 10 fixed workers is almost 7–8 s on a single machine. If we apply the same algorithm in a network of systems in a distributed fashion using a parallel computing environment in each system, the above mentioned overhead will be surely negligible.

3.7. Comparison with [99–101]

In the past, several authors have explored feature selection in parallel computing environments for network traffic analysis. However, our approach in this paper differs significantly from [99,100], and [101]. In Table 10, we show a comparison of our method with these methods.

4. Active learning in network traffic classification

Network traffic identification and classification are important areas of research for network security, network design, network monitoring and management. Due to the increasing size of transferred data and the diversity of applications available, it is necessary to understand the structure and the dynamics of networks, perform flow prioritization and diagnostic monitoring. Such activities are helped by traffic analysis, including classification.

A real time traffic classification approach has the potential to provide solutions for different network management problems, which are difficult to solve using statistical approaches. Every ML algorithm usually has a different approach to deal with features, leading to different dynamic behaviors during training and classification.

In network traffic classification, one interesting and yet challenging issue is the variety or spread of information. Spread of information means that every type of traffic can have unique characteristics or statistical properties. The set of nodes and connections between them in real network infrastructure is known, but the information about traffic characteristics can be sparse and not coherent. Furthermore, in large-scale networks, direct information about the label for a specific traffic instance is missing. All labels can be obtained by querying, but due to the scale of the network and anonymity issues, it is very costly.

Table 10
Comparison with [99–101].

Method & Year	Key feature(s)	Theoretical proof	Dataset(s) used	Performance
Bul'ajoul et al. [99], 2014	Utilization of QoS and parallel technologies	No	Traffic generated by NetScanPro[a] Tool	40,000 packets are processed. Speed-up factor is around 3.
Chen et al. [100], 2015	Horizontal and vertical data compression model and MapReduce [102] based parallelization approach	No	Normal and attack traffic from KDD CUP 1999[b] and CMDC2012[c]	8,000 instances are processed, speed-up factor is 4 to 8.
Kuttranont et al. [101], 2017	Integration of a simplified neighborhood classification using the percentage instead of group rank	No	Normal and attack traffic from KDD CUP 1999	20% out of the 4,90,000 instances is processed, CPU execution time is 1,112 to 1,116 s, and GPU execution time is 33 to 37 s. Speed-up factor is 3.
Our Method	Distributed feature selection in a number of horizontally partitioned datasets and a parallel cumulative ranker algorithm to rank the attributes	Yes	Normal and attack traffic from MIT-DARPA, CAIDA-2007, ISCX, and TUDDoS	50,000 to 1,000,000 records are processed, CPU execution time is around 66 to 84 s, and GPU execution time is around 10 to 24 s. Speed-up factor is 4 to 7.

[a] https://www.netscantools.com/nstpromain.html.
[b] http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.
[c] http://www.csmining.org/cdmc2012/index.php?id=14.

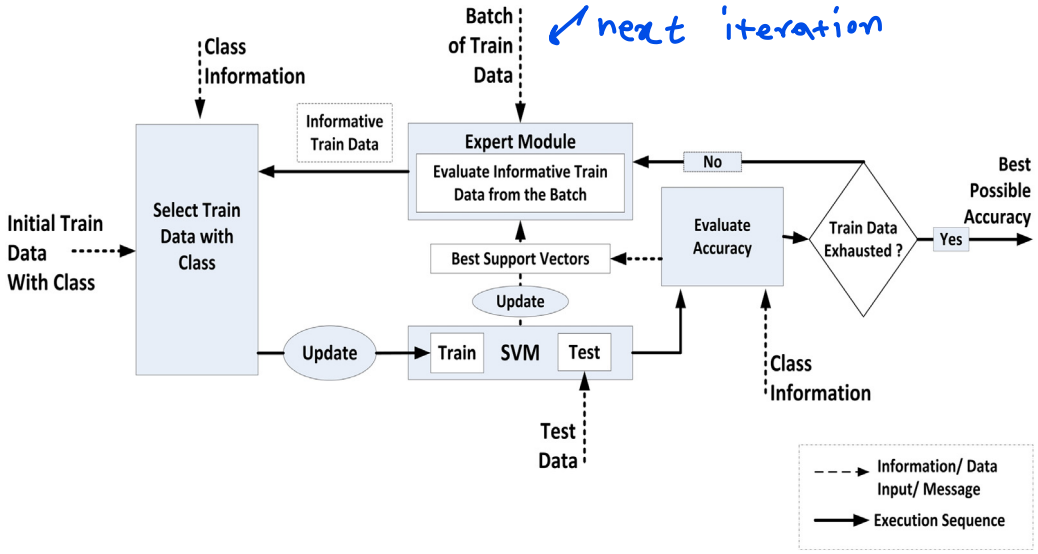


Fig. 13. Proposed Framework.

In the context of this paper, we assume that in a large network infrastructure, there is a huge surge in network traffic during a DDoS event. However, the resources to respond to this increased traffic are fixed, and are not sufficient. So we need to optimize our response. We have to classify the traffic to separate legitimate and illegitimate traffic using collective classification.

Collective classification refers to the classification of a set of interlinked objects using all possible information we have [103,104]. In order to perform a collective classification task, it is necessary to retrieve class labels for an initial population of traffic instances and use these class labels in the next round of classification. Thus, before classifying the entire traffic, some selected instances need to be labeled with the required information and their belongingness towards different classes determined.

Based on the initial information, classification can be performed for all other remaining instances of network traffic in batches. The main issue is to determine which instances should be selected early for labeling. The obvious answer is that it should be started with some random nodes that have class labels or those nodes that estimate the whole network most accurately. This approach to solve the problem of which nodes' labels should be acquired in order to perform the collective classification involves active learning or active inference.

Active learning. Active learning is a special case of semi-supervised machine learning in which a learning algorithm is able to interactively query the user (or some other information source) to obtain the desired outputs at new data points. It is also called optimal experimental design. There are situations in which unlabeled data is abundant but manually labeling such data is expensive. In such a scenario, learning algorithms can actively query the user, the teacher or the expert for labels. This type of iterative supervised learning is called active learning. Since the learner chooses the examples, the number of examples to learn a concept can often be much lower than the number required in normal supervised learning. With this approach, there is also a risk that the algorithm may be overwhelmed by uninformative examples.

Definition:. Let A be the total set of data under consideration. In the case of a DDoS attack, the DDoS traffic is embedded in the overall traffic. Thus, A would include all instances of network traffic that are known to have a certain malign activity and all additional instances

that one might want to test for that activity. During each iteration i, A is broken up into three subsets.

1. $A_{k,i}$: Data points for which the label is known.
2. $A_{u,i}$: Data points for which the label is unknown.
3. $A_{c,i}$: A subset of $A_{u,i}$, that needs to be labeled based on learning from the training data.

Most current research in active learning involves the best method on how to choose the data points for $A_{c,i}$.

4.1. Active learning framework

We develop an active learning approach to network traffic classification for detection of DDoS attack using appropriate training samples, identified by an expert module from a pool of unlabeled data in a batch mode.

4.2. Framework

Fig. 13 shows our proposed framework as a prototype. The whole process can be visualized in terms of various components, as discussed below (see Fig. 14).

4.2.1. Data components

We use four datasets, viz., MIT-DARPA,[7] CAIDA-2007,[8] ISCX,[9] and TUDDoS.[10] We choose five attributes of network traffic: source IP address, destination IP address, source port number, destination port number and frame length of each traffic sample, the same as before. Using *editcap*[11] and *Tshark*,[12] we obtain the values of these attributes from traffic data containing about 1,000,000 packets. The source IP addresses and destination IP addresses have been converted into decimal values.

1. **Training Data**: Training data are comprised of both normal and DDoS traffic instances, described using five attributes. The label for each instances comes from the pool of labels. These data are used to train an SVM binary classifier.
2. **Test data**: Test data are also comprised of both normal and DDoS traffic instances. The labels of each instance is in the pool of labels. The trained SVM classifier classifies the test data.
3. **Pool of labels**: In our framework, we keep all the labels of training and test data in a separate pool and for every instance of training and test data, there is an exact mapping of labels without any error. Labels for training data are given to the expert module, when needed to provide the appropriate instances to update the training data for the SVM classifier.
4. **Best Possible Support Vectors**: This data component is a set of support vectors, which is updated iteratively. After complete exhaustion of the training data, this component has the best set of support vectors to classify the test data.

4.2.2. Process components

For active learning, we have developed a few core processing components as per the framework.

A. **Selection of training data:** To initiate the active learning process, we select a few random instances of traffic data from the training data. Learning from these instances, the SVM computes n support vectors. Using these vectors, the expert module provides the appropriate training samples with class labels from the pool of labels on request from this component. $3 * n$ instances from the training data are evaluated in the expert system in each iteration. The appropriate instances are updated by this component during the training stage of the SVM.

B. **Expert module:** The number of samples in a batch of training data is thrice the total number support vectors generated in the previous round of SVM training of the classifier. The expert module processes the batch of training data to find the required samples according to the strategy discussed below. Chosen samples of data and classes are merged with the previous set of training data, for further training in the next round. Thus, we can get better boundary lines and an updated set of support vectors.

Algorithm 1 *ActiveRD_1*

1: **procedure** FindAccuracy
2: Select a few training samples, $TrainSet_i$.
3: Train SVM using $TrainSet_i$.
4: Test SVM on test data.
5: $True\_accuracy_i = 100 - (\frac{false_{labelling}}{total_{testlabel}} * 100)$.
6: $BestAccuracy = 0$
7: **if** $True\_accuracy_i > BestAccuracy$ **then**
8: BestAccuracy $= True\_accuracy_i$.
9: $BestS\_VectorSet = SupportVectorSet_i$
10: **if** Training Data Exhausted **then**
11: Stop $FindAccuracy$
12: **else**
13: Input $BestS\_VectorSet$ to $Expert\_Process$.
14: Call $Expert\_Process$ for next informative batch of training samples, $TrainSet_{i+1}$.
15: Go to Step 3.

$$True\_accuracy_i = 100 - (\frac{false_{labelling}}{total_{testlabel}} * 100) \quad (3)$$

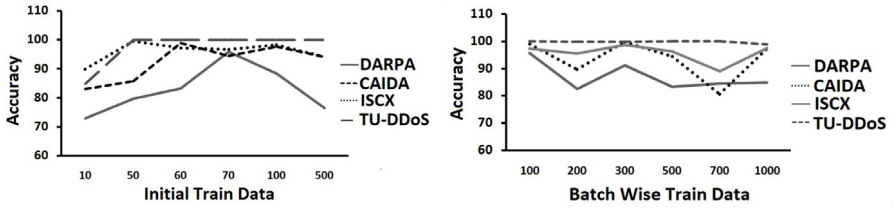
Algorithm 2 *ActiveRD_2*

1: **procedure** $Expert\_Process$
2: **if** Size of $BestS\_VectorSet$ = n **then**
3: Pool $3 * n$ training data samples as a batch from original training set.
4: Calculate $Euclid\_Dist$ for each support vector in $BestS\_VectorSet$ with $3 * n$ train data samples.
5: Choose training samples with maximum and minimum $Euclid\_Dist$ from each support vector from $BestS\_VectorSet$, to obtain $2 * n$ or fewer samples.
6: Input these training samples as $TrainSet_i$ to procedure $FindAccuracy$
7: Stop

Expert Module Strategy

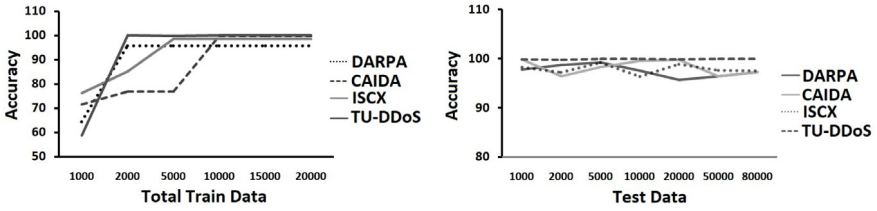
- Let n be the number of support vectors generated in the $(i-1)th$ training iteration of the SVM classifier.
- A batch of size $3 * n$ is pooled from the main training dataset for further evaluation in the expert module to find appropriate instances
- From each support vector provided by the SVM classifier, the Euclidean distance is computed to every sample in the batch. Samples with the shortest and longest distances from each support vector are selected to merge with the training data for the SVM classifier, to create the training data for the ith iteration. So, the number of strategically chosen samples is less than or equal to $2 * n$. For these chosen samples, we use the original classes/labels

[7] https://www.ll.mit.edu/ideval/data/.
[8] http://www.caida.org/data/.
[9] http://www.unb.ca/cic/datasets/ids.html.
[10] http://agnigarh.tezu.ernet.in/~dkb/tucannon/index.html.
[11] https://www.wireshark.org/docs/man-pages editcap.html.
[12] http://www.wireshark.org/docs/man-pages/tshark.html.



(a) Accuracy vs. Number of Initial Training Data Instances for All Datasets

(b) Accuracy vs. Number of Training Data Instances in a Batch for All Datasets



(c) Accuracy vs. Total Number of Training Data Instances for All Datasets

(d) Accuracy vs. Number of Total Test Data Instances for All Datasets

Fig. 14. Accuracies for Specific Scenarios as per Tables 11–14.

stored in Pool of labels. The Euclidean distance can be substituted with other distance metrics.

C. **SVM Classifier:** The SVM Classifier has two stages: training and testing. The selected or updated training data are used for learning and construction of the hyperplanes for classification are performed in this training phase. The generated support vectors are provided to the expert module for appropriate selection of instances, which are required for the next iteration of learning. SVM classifier classifies the test data using the hyperplanes generated in the learning phases.

D. **Evaluate Accuracy:** Variations between the classified class labels and the actual class labels are used to compute the accuracy of the proposed model. Labels on test data are required for finding the classification accuracy using Eq. (3) of our framework, when the SVM classifier classifies the test data into two separate classes.

4.3. Result

We use the datasets mentioned earlier and the following environment to carry out the experiments for the evaluation of the framework. The set-up has been described in Section 3.6.

In Tables 11–14, we present classification accuracies for the datasets under different conditions. Table columns are initial training data (A), batch-wise training data (B), total training data (C), total test data (D), total number of iterations (E), the highest accuracy (F) and the best support vectors (G). In the tables, we change the conditions of four columns: initial training data, batch-wise training data, total amount of training data, and total amount of test data one by one, and find the effect on classification accuracy.

In Table 11, we change the initial instances of training data (first iteration) for all datasets. We do so to find the number of initial instances to obtain high accuracy. A small number of initial instances that give high accuracy makes the active learning process more efficient.

In Table 12, we keep the number of initial instances constant to obtain higher accuracy, but we change the number of instances in the batch for the next iteration onwards. If a lower number of instances in a batch can give higher accuracy than a higher number of instances in the batch, the expert module works less to find the appropriate instances using support vectors for further classification.

However, a higher number of instances in a batch makes for fewer iterations to run through the whole train dataset. So, in Table 13, we change the total number of the training instances. A perfect tuning between number of instances in a batch and total number of instances in train dataset can reduce the iterations to run through the whole training data. In these four tables, we find how many instances in the training data can give higher accuracy with fewer iterations.

In Table 14, we keep the first three columns fixed at the specific settings for which we get higher accuracies in the previous tables. However, we change the number of instances in the test data. This is because, we want to see how the accuracy changes for a higher number of instances in the test data. In these tables, we see that after the tuning of the first three columns depending on the results of the previous tables, the accuracies do not vary much with variations in the number of instances in the test data (lower to higher), which is desired form the perspective of active learning.

In Tables 11–14, we have a column for the number of best support vectors which gives the best accuracies in a specific iteration. The number of support vectors does not depend upon any conditions or changes in the first three columns. After fixing the values of the first three columns at a specific setting, we get the actual support vectors for best classification accuracies.

Table 15 shows the difference between execution times in the sequential environment and in the parallel environment. The parallel environment is created using GPU worker nodes, as discussed in Section 3.6. The version of MATLAB we use has a parallel computing toolbox. We have used 10 workers in the work environment to run our evaluation process concurrently. These workers are MATLAB computational engines that run locally. The entire experiment is performed with workers present in a single machine in a distributed model using the parallel computing toolbox. We assume that the communication time required to send the traffic data to a single worker node of the GPU from the CPU is constant. In this table, we see that executing the same classification procedure in a parallel environment can reduce the execution time by 15% - 25% compared to the serial execution environment

4.4. Discussion

Based on the experiments reported in this paper, we enumerate our observations as follows.

- Our experiments demonstrate that a classification technique supported by an active learning approach is effective for network traffic classification.

Table 11
Classification accuracies vs. initial number of training instances for all datasets.

Datasets	A	B	C	D	E	F	G
DARPA	10	100	10,000	20,000	101	72.98	5
	50					79.71	7
	60					83.25	5
	70					95.76	5
	100					88.35	13
	500				97	76.44	23
CAIDA	10	100	10,000	20,000	101	83.08	10
	50					85.66	23
	60					98.79	7
	70					94.4	42
	100					97.57	53
	500				97	94.13	157
ISCX	10	100	10,000	20,000	101	89.88	14
	50					99.46	29
	60					97.19	7
	70					96.57	35
	100					98.21	74
	500				97	94.13	36
TU-DDoS	10	100	10,000	20,000	101	84.76	11
	50					99.91	20
	60					99.96	26
	70					99.95	37
	100					99.91	16
	500				97	99.96	15

For Tables 11 to 14: A. Initial Training Data Instances, B. Number of Training Data Instances in a Batch, C. Total Number of Training Data Instances, D. Number of Test Data Instances, E. Total Number of Iterations, F. Highest Accuracy, G. Number of Best Support Vectors of Instances.

Table 12
Classification accuracies vs. number of training instances in a batch for all datasets.

Datasets	A	B	C	D	E	F	G
DARPA	70	100	10,000	20,000	101	95.76	5
		200			51	82.48	10
		300			35	91.07	8
		500			21	83.27	10
		700			16	84.45	9
		1,000			11	84.87	10
CAIDA	60	100	10,000	20,000	101	98.79	7
		200			51	89.65	35
		300			35	99.79	9
		500			21	94.4	42
		700			16	80.57	38
		1,000			11	97.37	38
ISCX	50	100	10,000	20,000	101	97.19	7
		200			51	95.47	83
		300			35	98.61	33
		500			21	96.32	27
		700			16	88.92	34
		1,000			11	97.37	48
TU-DDoS	60	100	10,000	20,000	101	99.96	26
		200			51	99.87	17
		300			35	99.91	29
		500			21	99.99	19
		700			16	99.95	11
		1,000			11	98.76	11

- Classification accuracy is largely dependent on variations in *(i)* the number of instances in a batch, *(ii)* the initial number of instances and *(iii)* the total number of instances in training data.
- The number of iterations needed to identify the best possible support vectors is also dependent on the above variations.
- From a given training set of instances, we can extract the best possible support vectors for maximum accuracy for the given test data with the help of active learning.
- We can extract informative samples using the expert module in an unsupervised manner.
- The active learning approach has been found to perform consistently well with high accuracy for several datasets.

4.5. Comparison with [105] ,[40,106] and [107]

There is very little work on active learning based or semi-supervised network traffic analysis. Our approach in this paper differs significantly from the approaches discussed in [40,105,106] and [107]. In Table 16, we show a comparison of our method with these methods.

5. Case Study: Detection of DDoS Attack in Electric Power System

It is instructive and useful to see how our proposed framework works in a real life scenario. We have acquired 15 datasets on Power System Attacks provided by Mississippi State University and Oak Ridge

Table 13
Classification accuracies vs. total number of training instances for all datasets.

Datasets	A	B	C	D	E	F	G
DARPA	70	100	1,000	20,000	11	64.41	10
			2,000		21	95.76	5
			5,000		51	95.76	5
			10,000		101	95.76	5
			15,000		151	95.76	5
			20,000		201	95.76	5
CAIDA	60	300	1,000	20,000	5	71.56	40
			2,000		8	76.92	43
			5,000		18	76.92	43
			10,000		35	99.79	9
			15,000		51	99.79	9
			20,000		68	99.79	9
ISCX	50	300	1,000	20,000	5	76.17	40
			2,000		8	85.17	49
			5,000		18	98.61	33
			10,000		35	98.61	33
			15,000		51	98.61	33
			20,000		68	98.61	33
TU-DDoS	60	700	1,000	20,000	3	58.88	23
			2,000		4	99.99	18
			5,000		9	99.91	9
			10,000		16	99.95	11
			15,000		23	99.95	11
			20,000		30	99.95	11

Table 14
Classification accuracies vs. total number of test instances for all datasets.

Datasets	A	B	C	D	E	F	G
DARPA	70	100	2,000	1,000	21	97.8	5
				2,000		98.78	
				5,000		99.28	
				10,000		97.58	
				20,000		95.76	
				50,000		96.39	
				80,000		97.25	
CAIDA	60	300	10,000	1,000	35	99.85	9
				2,000		96.45	
				5,000		98.28	
				10,000		99.58	
				20,000		99.79	
				50,000		96.39	
				80,000		97.25	
ISCX	50	300	5,000	1,000	18	98.24	33
				2,000		97.27	
				5000		99.17	
				10,000		96.43	
				20,000		98.84	
				50,000		97.64	
				80,000		97.53	
TU-DDoS	60	700	2,000	1000	4	99.87	18
				2,000		99.82	
				5,000		99.95	
				10,000		99.94	
				20,000		99.91	
				50,000		99.97	
				80,000		99.95	

Table 15
Execution time in sequential computation Vs. parallel computation.

Datasets	Initial number of training data instances	Number of training data instances in a batch	Total number of training data instances	Test data	Total number of iterations	Highest accuracy	Execution time for serial computation (in sec.)	Execution time for parallel computation (in sec.)
DARPA	70	100	2,000	80,000	21	95.47	195.522	8.27
CAIDA	60	300	10,000		35	97.25	318.417	15.354
ISCX	50	300	5,000		18	97.53	187.243	8.772
TU-DDoS	60	700	2,000		4	99.95	40.798	2.407

National Laboratory [108]. In the network diagram as shown in Fig. 15, we have several components. Firstly, G1 and G2 are power generators. R1 through R4 are Intelligent Electronic Devices (IEDs) that can switch the breakers on or off. These breakers are labeled BR1 through BR4.

Table 16
Comparison with [105],[106],[40] and [107].

Method & Year	Key feature(s)	Dataset(s) used	Performance	Parallel execution
Jadidi et al. 2015 [107]	Use of SVM for semisupervised approach	Data from two data centers based on Citrix XenServer[a] and VMware ESXi[b], DARPA and CAIDA	94.01% accuracy is obtained with 129,571 flows	No
Haddadi and Zincir-Heywood, 2016 [105]	Use of traffic flow exporters and protocol filters	Zeus dataset[c] from the Snort web site and one Citadel, one Kelihos, and one Cutwail traffic data sets from the NETRESEC repository[d]	99.9% detection rate obtained with 10,000 flow instances	No
Ashfaq et al. 2017 [106]	Fuzziness based semi-supervised learning approach	NSL-KDD dataset[e]	Maximum accuracy obtained 84% with almost 50,000 records	No
Cao et al. 2017 [40]	Use of SVM training model called SPP-SVM and principal component analysis (PCA)	Andrew Moore dataset[f]	98.6% accuracy is obtained with 24,897 samples	No
Our Method	Use of SVM binary classifier for iterative active learning process	DARPA, CAIDA, ISCX and TU-DDoS datasets used	Maximum 99.9% accuracy obtained with 100,000 samples	Yes

[a] https://www.citrix.co.in/products/xenserver/.
[b] https://www.vmware.com/products/esxi-and-esx.html.
[c] http://www.unb.ca/cic/datasets/botnet.html.
[d] http://www.netresec.com.
[e] http://www.unb.ca/cic/datasets/nsl.html.
[f] https://researchportal.hw.ac.uk/en/persons/andrew-j-moore/datasets/.

Table 17
Power system attack dataset structure.

Scenario	Events	Class type
1	Fault from 10%–19% on L1	Natural Events
2	Fault from 20%–79% on L1	
3	Fault from 80%–90% on L1	
4	Fault from 10%–19% on L2	
5	Fault from 20%–79% on L2	
6	Fault from 80%–90% on L2	
7	Fault from 10%–19% on L1 with tripping command	Attack Events
8	Fault from 20%–79% on L1 with tripping command	
9	Fault from 80%–90% on L1 with tripping command	
10	Fault from 10%–19% on L2 with tripping command	
11	Fault from 20%–79% on L2 with tripping command	
12	Fault from 80%–90% on L2 with tripping command	
13	Line L1 maintenance	Natural Events
14	Line L2 maintenance	
15	Command Injection to R1	Attack Events
16	Command Injection to R2	
17	Command Injection to R3	
18	Command Injection to R4	
19	Command Injection to R1 and R2	
20	Command Injection to R3 and R4	
21	Fault from 10%–19% on L1 with R1 disabled & fault	
22	Fault from 20%–90% on L1 with R1 disabled & fault	
23	Fault from 10%–49% on L1 with R2 disabled & fault	
24	Fault from 50%–79% on L1 with R2 disabled & fault	
25	Fault from 80%–90% on L1 with R2 disabled & fault	
26	Fault from 10%–19% on L2 with R3 disabled & fault	
27	Fault from 20%–49% on L2 with R3 disabled & fault	
28	Fault from 50%–90% on L2 with R3 disabled & fault	
29	Fault from 10%–79% on L2 with R4 disabled & fault	
30	Fault from 80%–90% on L2 with R4 disabled & fault	
35	Fault from 10%–49% on L1 with R1 and R2 disabled & fault	
36	Fault from 50%–90% on L1 with R1 and R2 disabled & fault	
37	Fault from 10%–49% on L1 with R3 and R4 disabled & fault	
38	Fault from 50%–90% on L1 with R3 and R4 disabled & fault	
39	L1 maintenance with R1 and R2 disabled	
40	L1 maintenance with R1 and R2 disabled	
41	Normal operation load changes	No Event Scenarios

Table 18
Execution results using active learning to detect intrusion using ranked Feature.

Datasets	Active learning parameters		Number of features in optimal feature set	Features in optimal feature set	Accuracy (10-Fold cross validated)
	Number of initial samples	Number of samples in batch			
1			3	61, 92, 48	92.94
2			3	48, 77, 28	94.26
3			3	115, 63, 11	99.19
4			2	115, 48	99.45
5			2	28, 98	99.61
6			2	115, 65	97.98
7			5	48, 77, 115, 5, 69	90.63
8	10	100	2	26, 84	97.60
9			11	77, 65, 115, 48, 26, 61, 11, 15, 94, 57, 5	90.61
10			3	115, 77, 98	99.47
11			3	92, 115, 77	83.20
12			2	115, 106	96.73
13			12	27, 57, 15, 65, 28, 104, 106, 48, 92, 63, 42, 11	89.56
14			9	67, 65, 92, 115, 11, 98, 38, 69, 106	81.90
15			15	56, 115, 94, 26, 72, 84, 67, 107, 128, 42, 35, 65, 11, 17, 21	99.71

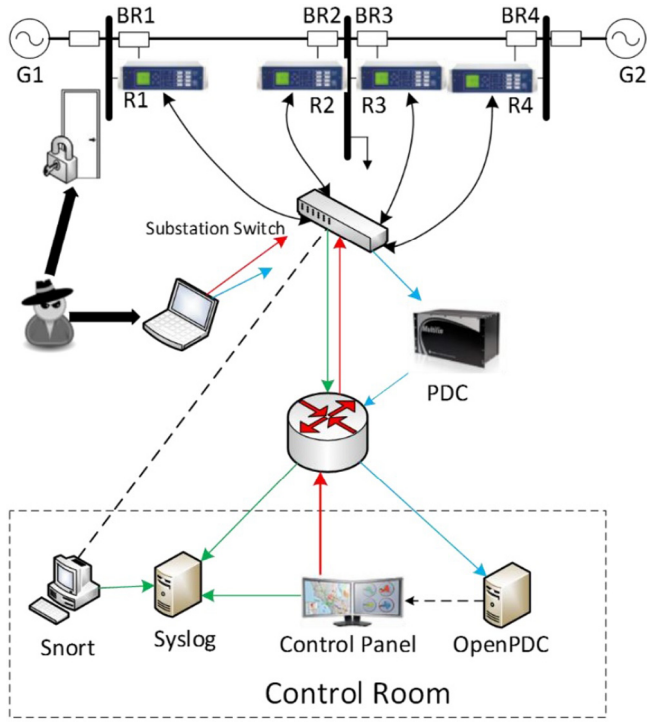


Fig. 15. Power System Framework Configuration Provided by Mississippi State University and Oak Ridge National Laboratory.

We also have two lines. Line One (L1) spans from Breaker One (BR1) to Breaker Two (BR2), and Line Two (L2) spans from Breaker Three (BR3) to Breaker Four (BR4). Each IED automatically controls one breaker. R1 controls BR1, R2 controls BR2, and so on. The IEDs use a distance protection scheme which trips the breaker on detected faults, whether actual or faked, they have no internal validation mechanism to detect the difference. Operators can also manually issue commands to the IEDs R1 through R4 to manually trip the breakers BR1 through BR4, respectively. The manual override is used when performing maintenance on the lines or other system components.

Power system attack datasets:. These datasets consist of fifteen sets with 37 power system event scenarios in each. The 37 scenarios are divided into Natural Events (8), No Events (1) and Attack Events (28). For our experimental purpose, we consider only the two-class problem (Natural and Attack) as shown in Table 17. Types of scenarios include the following.

1. Short-circuit fault: This is a short in a power line and can occur in various locations along the line. The location is indicated by the percentage range.
2. Line maintenance: One or more relays are disabled on a specific line to do maintenance for that line.
3. Remote tripping command injection (Attack): This is an attack that sends a command to a relay which causes a breaker to open. It can only be done once an attacker has penetrated outside defenses.
4. Relay setting change (Attack): Relays are configured with a distance protection scheme and the attacker changes the setting to disable the relay function such that relay will not trip for a valid fault or a valid command.
5. Data Injection (Attack): Here, a valid fault by changing values to parameters such as current, voltage and sequence components are imitated. This attack aims to blind the operator and causes a black-out.

There are approximately 129 features in every dataset. There are 29 types of measurements from each phasor measurement unit (PMU). A phasor measurement unit (PMU) or synchrophasor is a device which measures the electrical waves on an electricity grid, using a common time source for synchronization. In addition, there are 4 PMUs which measure 29 features for 116 PMU measurement columns. Further, there are 12 columns for control panel logs, Snort alerts and relay logs of the 4 PMUs/relays (relay and PMU are integrated together). The last column is the marker, and so in there are total 129 columns in each dataset.

5.1. Results

We combine our frameworks of feature ranking and active learning to discover anomalies in these datasets. The results are presented in Table 18.

6. Conclusion

Classification of anomalous traffic is difficult in current networks and systems, especially when it is necessary to deal with a large volume of traffic instances. We have presented a method to rank traffic features to support the selection of a best subset of features to achieve high classification accuracy. To handle large volumes of network traffic, we present a parallel computation approach. In addition to theoretical evidence, our experimental results are convincing from both execution time performance as well as classification accuracy. Further, we uncover interesting findings regarding the interaction of worker nodes and dataset partitions with classification accuracy. PCR can be extended by incorporating soft computing and other techniques to enable ranking of features for critical voluminous data spaces to establish its generality. Work is underway towards development of a fuzzy reasoning enabled PCR supported by active learning.

We have presented a method to learn from traffic in batches to label test instances to maintain high classification accuracy. To handle voluminous network traffic, we use the active learning approach with fewer training samples. We justify the suitability of this approach. In addition to theoretical evidence, our experimental results are convincing from classification accuracy. Our frameworks work well on power system attack datasets.

Declaration of competing interest

The authors have affiliations with organizations with direct or indirect financial interest in the subject matter discussed in the manuscript.

References

[1] D.E. Denning, An intrusion-detection model, IEEE Trans. Softw. Eng. (2) (1987) 222–232.

[2] V. Bolón-Canedo, N. Sánchez-Maroño, A. Alonso-Betanzos, Recent advances and emerging challenges of feature selection in the context of big data, Knowl.-Based Syst. 86 (2015) 33–45.

[3] M. Roesch, Snort–the de facto standard for intrusion detection/prevention, 2005, http://www.snort.org, online. (Accessed 15 January 2017).

[4] R.A. Kemmerer, G. Vigna, Intrusion detection: A brief history and overview, Computer 35 (4) (2002) supl27–supl30.

[5] L. Stewart, G. Armitage, P. Branch, S. Zander, An architecture for automated network control of QoS over consumer broadband links, in: TENCON 2005 2005 IEEE Region 10, IEEE, 2005, pp. 1–6.

[6] R. Deka, D. Bhattacharyya, J. Kalita, DDoS attacks: Tools, mitigation approaches, and probable impact on private cloud environment, 2017, arXiv preprint arXiv:1710.08628.

[7] G. Aceto, A. Dainotti, W. De Donato, A. Pescapé, Portload: Taking the best of two worlds in traffic classification, in: INFOCOM IEEE Conference on Computer Communications Workshops, 2010, IEEE, 2010, pp. 1–5.

[8] L. Bernaille, R. Teixeira, K. Salamatian, Early application identification, in: Proceedings of the 2006 ACM CoNEXT Conference, ACM, 2006, p. 6.

[9] W. Li, M. Canini, A.W. Moore, R. Bolla, Efficient application identification and the temporal and spatial stability of classification schema, Comput. Netw. 53 (6) (2009) 790–809.

[10] J. Zhang, X. Chen, Y. Xiang, W. Zhou, J. Wu, Robust network traffic classification, IEEE/ACM Trans. Netw. 23 (4) (2015) 1257–1270.

[11] J. Zhang, Y. Xiang, W. Zhou, Y. Wang, Unsupervised traffic classification using flow statistical properties and IP packet payload, J. Comput. System Sci. 79 (5) (2013) 573–585.

[12] A. Tongaonkar, R. Torres, M. Iliofotou, R. Keralapura, A. Nucci, Towards self adaptive network traffic classification, Comput. Commun. 56 (2015) 35–46.

[13] M. Shafiq, X. Yu, D. Wang, Network traffic classification using machine learning algorithms, in: International Conference on Intelligent and Interactive Systems and Applications, Springer, 2017, pp. 621–627.

[14] D.A. Cohn, Z. Ghahramani, M.I. Jordan, Active learning with statistical models, J. Artif. Intell. Res. (1996).

[15] S. Tong, D. Koller, Support vector machine active learning with applications to text classification, J. Mach. Learn. Res. 2 (Nov) (2001) 45–66.

[16] K. Brinker, Incorporating diversity in active learning with support vector machines, in: Proceedings of the 20th International Conference on Machine Learning, ICML-03, 2003, pp. 59–66.

[17] R. Yuan, Z. Li, X. Guan, L. Xu, An SVM-based machine learning method for accurate internet traffic classification, Inf. Syst. Front. 12 (2) (2010) 149–156.

[18] H. Liao, L. Chen, Y. Song, H. Ming, Visualization-based active learning for video annotation, IEEE Trans. Multimed. 18 (11) (2016) 2196–2205.

[19] A. Gupta, R. Gutierrez-Osuna, M. Christy, R. Furuta, L. Mandell, Font identification in historical documents using active learning, 2016, arXiv preprint arXiv:1601.07252.

[20] J. Morgan, Streaming network traffic analysis using active learning, 2015.

[21] J. Morgan, A.N. Zincir-Heywood, J.T. Jacobs, A benchmarking study on stream network traffic analysis using active learning, in: Recent Advances in Computational Intelligence in Defense and Security, Springer, 2016, pp. 249–273.

[22] J. Zhang, C. Chen, Y. Xiang, W. Zhou, A.V. Vasilakos, An effective network traffic classification method with unknown flow detection, IEEE Trans. Netw. Serv. Manag. 10 (2) (2013) 133–147.

[23] J. Zhang, Y. Xiang, Y. Wang, W. Zhou, Y. Xiang, Y. Guan, Network traffic classification using correlation information, IEEE Trans. Parallel Distrib. Syst. 24 (1) (2013) 104–117.

[24] B. Luo, J. Xia, A novel intrusion detection system based on feature generation with visualization strategy, Expert Syst. Appl. 41 (9) (2014) 4139–4147.

[25] F. Kuang, W. Xu, S. Zhang, A novel hybrid KPCA and SVM with GA model for intrusion detection, Appl. Soft Comput. 18 (2014) 178–184.

[26] A.S. Eesa, Z. Orman, A.M.A. Brifcani, A novel feature-selection approach based on the Cuttlefish optimization algorithm for intrusion detection systems, Expert Syst. Appl. 42 (5) (2015) 2670–2679.

[27] L. Xiao, Y. Chen, C.K. Chang, Bayesian model averaging of Bayesian network classifiers for intrusion detection, in: 38th International Computer Software and Applications Conference Workshops, COMPSACW, 2014, IEEE, 2014, pp. 128–133.

[28] P. Mitra, B.U. Shankar, S.K. Pal, Segmentation of multispectral remote sensing images using active support vector machines, Pattern Recognit. Lett. 25 (9) (2004) 1067–1074.

[29] R. Pang, M. Allman, M. Bennett, J. Lee, V. Paxson, B. Tierney, A first look at modern enterprise traffic, in: Proceedings of the 5th ACM SIGCOMM Conference on Internet Measurement, 2005, USENIX Association, 2005, p. 2.

[30] Y. Wang, Y. Xiang, J. Zhang, S. Yu, A novel semi-supervised approach for network traffic clustering, in: 5th International Conference on Network and System Security, NSS, 2011, IEEE, 2011, pp. 169–175.

[31] U. Ravale, N. Marathe, P. Padiya, Feature selection based hybrid anomaly intrusion detection system using K means and RBF kernel function, Procedia Comput. Sci. 45 (2015) 428–435.

[32] S. Chawla, M. Sachdeva, S. Behal, Discrimination of DDoS attacks and flash events using Pearson's product moment correlation method, Int. J. Comput. Sci. Inf. Secur. 14 (10) (2016) 382.

[33] A.R. Yusof, N.I. Udzir, A. Selamat, An Evaluation on KNN-SVM Algorithm for Detection and Prediction of DDoS Attack, Springer, 2016, pp. 95–102.

[34] L. Bravi, V. Piccialli, M. Sciandrone, An optimization-based method for feature ranking in nonlinear regression problems, IEEE Trans. Neural Netw. Learn. Syst. 28 (4) (2017) 1005–1010.

[35] A. Lendasse, J.A. Lee, V. Wertz, M. Verleysen, Time series forecasting using CCA and Kohonen maps-application to electricity consumption, in: Proceedings of European Symposium on Artificial Neural Networks Bruges, Belgium, 2000, pp. 329–334.

[36] A. Lendasse, J. Lee, V. Wertz, M. Verleysen, Forecasting electricity consumption using nonlinear projection and self-organizing maps, Neurocomputing 48 (1) (2002) 299–311.

[37] B. Efron, T. Hastie, I. Johnstone, R. Tibshirani, et al., Least angle regression, Ann. Statist. 32 (2) (2004) 407–499.

[38] U. Huebner, N. Abraham, C. Weiss, Dimensions and entropies of chaotic intensity pulsations in a single-mode far-infrared NH 3 laser, Phys. Rev. A 40 (11) (1989) 6354.

[39] A.S. Weigend, N.A. Gershenfeld, Results of the time series prediction competition at the Santa Fe Institute, in: International Conference on Neural Networks, 1993, IEEE, 1993, pp. 1786–1793.

[40] J. Cao, Z. Fang, G. Qu, H. Sun, D. Zhang, An accurate traffic classification model based on support vector machines, Int. J. Netw. Manage. 27 (1) (2017).

[41] A. Moore, D. Zuev, M. Crogan, Discriminators for use in flow-based classification, Technical Report, Dept. of Computer Science, Queen Mary, University of London, 2005.

[42] B. Jia, X. Huang, R. Liu, Y. Ma, A DDoS attack detection method based on hybrid heterogeneous multiclassifier ensemble learning, J. Electr. Comput. Eng. 2017 (2017).

[43] S. Rosset, A. Inger, KDD-cup 99: Knowledge discovery in a charitable organization's donor database, SIGKDD Explor. Newsl. 1 (2) (2000) 85–90, http://dx.doi.org/10.1145/846183.846204.

[44] C.F. Van Loan, Generalizing the singular value decomposition, SIAM J. Numer. Anal. 13 (1) (1976) 76–83.

[45] L. De Lathauwer, B. De Moor, J. Vandewalle, A multilinear singular value decomposition, SIAM J. Matrix Anal. Appl. 21 (4) (2000) 1253–1278.

[46] H. Kim, K.C. Claffy, M. Fomenkov, D. Barman, M. Faloutsos, K. Lee, Internet traffic classification demystified: Myths, caveats, and the best practices, in: Proceedings of the ACM CoNEXT Conference, 2008, ACM, 2008, p. 11.

[47] T. Karagiannis, A. Broido, N. Brownlee, K.C. Claffy, M. Faloutsos, Is p2p dying or just hiding?[p2p traffic measurement], in: Global Telecommunications Conference, 2004. GLOBECOM'04. IEEE, vol. 3, IEEE, 2004, pp. 1532–1538.

[48] T. Karagiannis, A. Broido, M. Faloutsos, et al., Transport layer identification of p2p traffic, in: Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement, ACM, 2004, pp. 121–134.

[49] T. Choi, C. Kim, S. Yoon, J. Park, B. Lee, H. Kim, H. Chung, T. Jeong, Content-aware internet application traffic measurement and analysis, in: Network Operations and Management Symposium, 2004. NOMS 2004. IEEE/IFIP, vol. 1, IEEE, 2004, pp. 511–524.

[50] V. Paxson, Empirically derived analytic models of wide-area TCP connections, IEEE/ACM Trans. Netw. 2 (4) (1994) 316–336.

[51] C. Dewes, A. Wichmann, A. Feldmann, An analysis of internet chat systems, in: Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement, ACM, 2003, pp. 51–64.

[52] S. Seufert, D. O'Brien, Machine learning for automatic defence against distributed denial of service attacks, in: Communications, 2007. ICC'07. IEEE International Conference on, IEEE, 2007, pp. 1217–1222.

[53] H. Dreger, A. Feldmann, M. Mai, V. Paxson, R. Sommer, Dynamic application-layer protocol analysis for network intrusion detection, in: USENIX Security Symposium, 2006, pp. 257–272.

[54] A.W. Moore, K. Papagiannaki, Toward the accurate identification of network applications, in: PAM, vol. 5, Springer, 2005, pp. 41–54.

[55] J. Sherry, C. Lan, R.A. Popa, S. Ratnasamy, Blindbox: Deep packet inspection over encrypted traffic, in: ACM SIGCOMM Computer Communication Review, vol. 45, ACM, 2015, pp. 213–226.

[56] C. Hu, H. Li, Y. Jiang, Y. Cheng, P. Heegaard, Deep semantics inspection over big network data at wire speed, IEEE Netw. 30 (1) (2016) 18–23.

[57] T. Lang, G. Armitage, P. Branch, H.-Y. Choo, A synthetic traffic model for half-life, in: Australian Telecommunications Networks & Applications Conference, vol. 2003, 2003.

[58] T. Lang, P. Branch, G. Armitage, A synthetic traffic model for quake3, in: Proceedings of the 2004 ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, ACM, 2004, pp. 233–238.

[59] P. Perera, Y.-C. Tian, C. Fidge, W. Kelly, A comparison of supervised machine learning algorithms for classification of communications network traffic, in: International Conference on Neural Information Processing, Springer, 2017, pp. 445–454.

[60] R. Raveendran, R.R. Menon, A novel aggregated statistical feature based accurate classification for internet traffic, in: Data Mining and Advanced Computing (SAPIENCE), International Conference on, IEEE, 2016, pp. 225–232.

[61] J. Park, H.-R. Tyan, C.-C. Kuo, Internet traffic classification for scalable QoS provision, in: Multimedia and Expo, 2006 IEEE International Conference on, IEEE, 2006, pp. 1221–1224.

[62] J. Erman, M. Arlitt, A. Mahanti, Traffic classification using clustering algorithms, in: Proceedings of the 2006 SIGCOMM Workshop on Mining Network Data, ACM, 2006, pp. 281–286.

[63] T. Auld, A.W. Moore, S.F. Gull, Bayesian neural networks for internet traffic classification, IEEE Trans. Neural Netw. 18 (1) (2007) 223–239.

[64] F. Hao, M. Kodialam, T. Lakshman, H. Song, Fast dynamic multiple-set membership testing using combinatorial Bloom filters, IEEE/ACM Trans. Netw. 20 (1) (2012) 295–304.

[65] V. Bolon-Canedo, N. Sanchez-Marono, A. Alonso-Betanzos, Feature selection and classification in multiple class datasets: An application to KDD Cup 99 dataset, Expert Syst. Appl. 38 (5) (2011) 5947–5957.

[66] Y. Saeys, I. Inza, P. Larrañaga, A review of feature selection techniques in bioinformatics, Bioinformatics 23 (19) (2007) 2507–2517.

[67] V. Bolón-Canedo, N. Sánchez-Marono, A. Alonso-Betanzos, J.M. Benítez, F. Herrera, A review of microarray datasets and applied feature selection methods, Inform. Sci. 282 (2014) 111–135.

[68] A. Jain, D. Zongker, Feature selection: Evaluation, application, and small sample performance, IEEE Trans. Pattern Anal. Mach. Intell. 19 (2) (1997) 153–158.

[69] B. Remeseiro, V. Bolon-Canedo, D. Peteiro-Barral, A. Alonso-Betanzos, B. Guijarro-Berdinas, A. Mosquera, M.G. Penedo, N. Sánchez-Marono, A methodology for improving tear film lipid layer classification, IEEE J. Biomed. Health Inf. 18 (4) (2014) 1485–1493.

[70] J. Yang, D. Zhang, X. Yong, J.-y. Yang, Two-dimensional discriminant transform for face recognition, Pattern Recognit. 38 (7) (2005) 1125–1129.

[71] S.H. Lee, J.Y. Choi, K.N. Plataniotis, Y.M. Ro, Color component feature selection in feature-level fusion based color face recognition, in: IEEE International Conference on Fuzzy Systems (FUZZ), 2010, IEEE, 2010, pp. 1–6.

[72] G. Forman, An extensive empirical study of feature selection metrics for text classification, J. Mach. Learn. Res. 3 (Mar) (2003) 1289–1305.

[73] S. Baccianella, A. Esuli, F. Sebastiani, Feature selection for ordinal text classification, Neural Comput. 26 (3) (2014) 557–591.

[74] D.J. Hand, H. Mannila, P. Smyth, Principles of Data Mining, MIT Press, Cambridge, Massachusetts, 2001.

[75] P. Lewis, The characteristic selection problem in recognition systems, IRE Trans. Inf. Theory 8 (2) (1962) 171–178.

[76] G.H. John, R. Kohavi, K. Pfleger, et al., Irrelevant features and the subset selection problem, in: Proceedings of the Eleventh International Conference Machine Learning, 1994, 1994, pp. 121–129.

[77] M. Dash, H. Liu, Feature selection for classification, Intell. Data Anal. 1 (1–4) (1997) 131–156.

[78] T.T. Nguyen, G. Armitage, A survey of techniques for internet traffic classification using machine learning, IEEE Commun. Surv. Tutor. 10 (4) (2008) 56–76.

[79] Y.-s. Lim, H.-c. Kim, J. Jeong, C.-k. Kim, T.T. Kwon, Y. Choi, Internet traffic classification demystified: On the sources of the discriminative power, in: Proceedings of the 6th International Conference on Co-NEXT, 2010, ACM, 2010, p. 9.

[80] M. Roughan, S. Sen, O. Spatscheck, N. Duffield, Class-of-service mapping for QoS: A statistical signature-based approach to IP traffic classification, in: Proceedings of the 4th ACM SIGCOMM Conference on Internet Measurement, 2004, ACM, 2004, pp. 135–148.

[81] Y. Xiang, W. Zhou, M. Guo, Flexible deterministic packet marking: An IP traceback system to find the real source of attacks, IEEE Trans. Parallel Distrib. Syst. 20 (4) (2009) 567–580.

[82] Z.M. Fadlullah, T. Taleb, A.V. Vasilakos, M. Guizani, N. Kato, DTRAB: Combating against attacks on encrypted protocols through traffic-feature analysis, IEEE/ACM Trans. Netw. 18 (4) (2010) 1234–1247.

[83] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility, Future Gener. Comput. Syst. 25 (6) (2009) 599–616.

[84] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, et al., A view of cloud computing, Commun. ACM 53 (4) (2010) 50–58.

[85] R.K. Deka, K.P. Kalita, D. Bhattacharya, J.K. Kalita, Network defense: Approaches, methods and techniques, J. Netw. Comput. Appl. 57 (2015) 71–84.

[86] P. Mitra, C. Murthy, S.K. Pal, Unsupervised feature selection using feature similarity, IEEE Trans. Pattern Anal. Mach. Intell. 24 (3) (2002) 301–312.

[87] J. Silva, A. Aguiar, F. Silva, A parallel computing hybrid approach for feature selection, in: 18th International Conference on Computational Science and Engineering, CSE, 2015, IEEE, 2015, pp. 97–104.

[88] F. Azmandian, A. Yilmazer, J.G. Dy, J.A. Aslam, D.R. Kaeli, GPU-accelerated feature selection for outlier detection using the local kernel density ratio, in: 12th International Conference on Data Mining, ICDM, 2012, IEEE, 2012, pp. 51–60.

[89] L. Morán-Fernández, V. Bolón-Canedo, A. Alonso-Betanzos, Centralized vs. distributed feature selection methods based on data complexity measures, Knowl.-Based Syst. 117 (2017) 27–45.

[90] S. Ramírez-Gallego, I. Lastra, D. Martínez-Rego, V. Bolón-Canedo, J.M. Benítez, F. Herrera, A. Alonso-Betanzos, Fast-MRMR: Fast minimum redundancy maximum relevance algorithm for high-dimensional big data, Int. J. Intell. Syst. 32 (2) (2017) 134–152.

[91] Y. Xu, Q. Zhu, Z. Fan, M. Qiu, Y. Chen, H. Liu, Coarse to fine K-nearest neighbor classifier, Pattern Recognit. Lett. 34 (9) (2013) 980–986.

[92] C. Liu, H. Wechsler, Gabor feature based classification using the enhanced Fisher linear discriminant model for face recognition, IEEE Trans. Image Process. 11 (4) (2002) 467–476.

[93] A.Y. Ng, M.I. Jordan, On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes, Adv. Neural Inf. Process. Syst. 2 (2002) 841–848.

[94] K.-Y. Feng, Y.-D. Cai, K.-C. Chou, Boosting classifier for predicting protein domain structural class, Biochem. Biophys. Res. Commun. 334 (1) (2005) 213–217.

[95] S.R. Safavian, D. Landgrebe, A survey of decision tree classifier methodology, IEEE Trans. Syst. Man Cybern. 21 (3) (1991) 660–674.

[96] J. Biesiada, W. Duch, Feature selection for high-dimensional data—A Pearson redundancy based filter, in: Computer Recognition Systems 2, Springer, 2007, pp. 242–249.

[97] S.-D. Bolboaca, L. Jäntschi, Pearson versus Spearman, Kendall's tau correlation analysis on structure-activity relationships of biologic active compounds, Leonardo J. Sci. 5 (9) (2006) 179–200.

[98] E. Frome, Algorithm as 171: Fisher's exact variance test for the Poisson distribution, J. R. Stat. Soc. Ser. C. Appl. Stat. 31 (1) (1982) 67–71.

[99] W. Bul'ajoul, A. James, M. Pannu, Improving network intrusion detection system performance through quality of service configuration and parallel technology, J. Comput. System Sci. 81 (6) (2015) 981–999.

[100] T. Chen, X. Zhang, S. Jin, O. Kim, Efficient classification using parallel and scalable compressed model and its application on intrusion detection, Expert Syst. Appl. 41 (13) (2014) 5972–5983.

[101] P. Kuttranont, K. Boonprakob, C. Phaudphut, S. Permpol, P. Aimtongkhamand, U. KoKaew, B. Waikham, C. So-In, Parallel KNN and neighborhood classification implementations on GPU for network intrusion detection, J. Telecommun. Electron. Comput. Eng. 9 (2–2) (2017) 29–33.

[102] J. Dean, S. Ghemawat, MapReduce: Simplified data processing on large clusters, Commun. ACM 51 (1) (2008) 107–113.

[103] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, T. Eliassi-Rad, Collective classification in network data, AI Mag. 29 (3) (2008) 93.
[104] M. McPherson, L. Smith-Lovin, J.M. Cook, Birds of a feather: Homophily in social networks, Annu. Rev. Sociol. 27 (1) (2001) 415–444.
[105] F. Haddadi, A.N. Zincir-Heywood, Benchmarking the effect of flow exporters and protocol filters on botnet traffic classification, IEEE Syst. J. 10 (4) (2016) 1390–1401.
[106] R.A.R. Ashfaq, X.-Z. Wang, J.Z. Huang, H. Abbas, Y.-L. He, Fuzziness based semi-supervised learning approach for intrusion detection system, Inform. Sci. 378 (2017) 484–497.
[107] Z. Jadidi, V. Muthukkumarasamy, E. Sithirasenan, K. Singh, Flow-based anomaly detection using semisupervised learning, in: Signal Processing and Communication Systems (ICSPCS), 2015 9th International Conference on, IEEE, 2015, pp. 1–5.
[108] Mississippi State University and Oak Ridge National Laboratory, Power system attack datasets, 2014, http://www.ece.uah.edu/~thm0009/icsdatasets/PowerSystemDatasetREADME.pdf.