IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 2, 2020, accepted March 15, 2020, date of publication March 26, 2020, date of current version April 9, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2983605

Online Data Center Traffic Classification Based on Inter-Flow Correlations

MERIEM AMINA SI SABER, MEHDI GHORBANI, ABDOLKHALEGH BAYATI, KIM-KHOA NGUYEN, AND MOHAMED CHERIET, (Senior Member, IEEE)
École de Technologie Supérieure (ÉTS), University of Quebec, Montreal, QC H3C 1K3, Canada

Corresponding author: Meriem Amina Si Saber (meriem-amina.si-saber.1@ens.etsmtl.ca)

This work was supported in part by the project CRDPJ 461084, in part by the Ciena, and in part by the National Sciences and Engineering Research Council of Canada (NSERC). The work of Mohamed Cheriet was supported by the Canada Research Chair, Tier 1.

ABSTRACT Today, increasing attention is being paid to Data Center (DC) traffic classification since these infrastructures have become the heart of a variety of time-sensitive and data-intensive service platforms. Classification provides the required tools for better understanding traffic patterns in order to ensure high Quality of Service (QoS) performances and solve scalability problems. Unfortunately, existing classification algorithms cannot deal efficiently with two critical challenges in DC traffic: inter-class imbalance and critical time constraints. In this paper, we propose a novel correlation-based algorithm following a cost-sensitive approach combined with a Bagged Random Forest (BRF) ensemble algorithm, to address the inter-class imbalance problem while meeting time requirements in a data center environment. In this strategy, a new method based on Reverse k-Nearest Neighbors (RkNN) is proposed to capture the rebalancing weights expressing inter-flow correlations, in order to perform an online classification approach. We demonstrate the efficiency of the algorithm by comparing its performance to several existing methods from data level, algorithm level, and cost-sensitive strategies on four real-world datasets. The results reveal that the proposed algorithm outperforms most approaches in the different datasets in terms of precision, recall, F1 measure, AUC and Kappa, as opposed to other algorithms that result in either high precision with low recall and low precision and high recall causing congestion or resource over provisioning.

INDEX TERMS Data center, network traffic classification, interflow correlation, ensemble algorithms, random forest, data imbalance.

I. INTRODUCTION

D ata centers (DC) are infrastructures holding large clusters of interconnected servers. Because of their rapid spread and the variety of services they can offer, from video on-demand, web searching, and gaming to storage and even computing, DCs have become the heart of the current digital world. DC traffic can be classified following two approaches into different classes: the first, a multi class classification, is application-based and identifies flows according to communication protocols; the second, the subject of this research, is binary and considers mainly flow sizes, it identifies traffic as *elephant* or *mice* [1].

Long-lived elephant flows account for a notable percentage of all the packets transmitted (80% of DC's total traffic), as opposed to mice flows, which contribute with a less important percentage but occur more frequently [2]. Elephant flows can cause buffer and link congestion, delaying latency-sensitive mice flows and thereby resulting in network performance degradation. Mice flows, on the other hand, are bursty and generate more control plane requests; this can cause jitter, which similarly degrades network Quality of Service (QoS) [3]. Optimal performance can then only be guaranteed by solving the trade-off equation between high throughput and low latency of both classes. Thus, a deeper understanding of traffic patterns can significantly improve the performance of the control mechanism, which results in adaptive/dynamic network QoS and also builds more efficient DCs in terms of resource consumption (energy, bandwidth, etc.) [4] and security (detect malicious traffic) [5].

When dealing with DC traffic, another problem arises. It is described as data imbalance [6] and appears between the small number of elephant flows (thus, representing the minority class) and the massive amount of mice flows (as the

The associate editor coordinating the review of this manuscript and approving it for publication was Yulei Wu.

majority class); Traffic classification becomes more challenging since machine learning algorithms will focus mainly on majority classes, decreasing classification performances and hence causing QoS degradation. Efficient rebalancing based classification strategies must be deployed in order to avoid congestion caused by false positives (elephant flows predicted as mice flows) or over provisioning caused by false negatives (mice flows predicted as elephant flows). Also, cloud providers need to quickly distinguish the classes transported by the massive amount of traffic flowing in their network in order to support their various Service Level Agreements (SLA). Indeed, when considering real-time network traffic scheduling, online classification is required as opposed to offline training, also known as late classification. Unlike offline flow classification, which extracts features upon the reception of all packets of a flow, online classification trains the algorithm from the first few packets only and provides early flow prediction while the flow is in progress. We also distinguish real-time identification from non-real-time, expressing fast or slow flow labeling during the testing phase of the algorithm.

Authors in [7] emphasize the impact of data imbalance in network traffic classification and on the fact that the previous works that resolve this problem are scarce. Indeed, most of the prior work on mice and elephant flow identification [3], [8] either assume the uniformity of traffic classes and thus do not consider the data imbalance problem but rather focus on enhancing traffic classification performances in ideal scenarios; or they classify only majority classes, because there approaches lack of rebalancing strategies and consequently fail to classify minority classes. Other algorithms [9], can not provide early classification and instead build offline methods since their algorithms do not guarantee efficient classification with only a few samples. Therefore, the most critical problem in DC traffic classification is to perform online flow classification when traffic is highly imbalanced because packet loss or disorder resulting from using few packets in the online classification process (also called packet reduction) can affect the performances of the classifiers in real-time networks [8] and imbalanced datasets may result in inefficient resource provisioning (congestion or over provisioning).

Several works emphasize the power of ensemble algorithms against data imbalance [10], and there again, two possible approaches can be deployed to enhance classification performances: combining ensemble algorithms with cost-sensitive learning or starting by rebalancing the classes distributions through sampling (oversampling, undersampling). While the former exclusively relies on the efficiency of the misclassification matrix and may suffer from weak accuracy for minority classes, the later may result in significant overhead and may not be adequate for online training.

To address this problem, we develop a novel Correlation-based Cost-Sensitive (CCS) algorithm. Instead of considering flows individually, we exploit the correlation among them; this increases the performances of the classification model, especially when there are not enough samples in the training set. More specifically, we propose to use inter-flow correlations as classification weights instead of penalty costs in the misclassification matrix, which brings the challenge of efficiently characterizing inter-flow correlations, since the flow is already defined as correlated packets (packets sharing five tuples: source and destination IP addresses and port number along with the protocol). Thus, it is crucial to find the correct correlation metrics. We propose to model inter-flow correlations as the distances between entering flows and the training set, following a radius based Reverse k Nearest Neighbor strategy, where we address the high dimension limitation of the algorithm using feature selection.

In this paper, we target online mice and elephant flow identification that are characterized by imbalanced distributions and propose a novel inter-flow correlation-based cost sensitive approach. The key contributions of this paper are:

- We model rebalancing weights as correlations between flows instead of misclassifications, building along with ensemble Bagged Random Forest (BRF), a novel cost sensitive oriented classification approach that addresses data imbalance differently.
- We propose a correlation-based cost-sensitive classification approach that alleviates the data imbalance problem while ensuring online traffic classification, using only eight first packets in each flow. This is based on the fact that the application synchronization phase at the beginning of each communication contains the required information to distinguish applications correctly.
- We propose an RkNN-based correlation classification strategy to measure the distance between new flows and bag of flows training the ensemble algorithm.
- We evaluate the efficiency of the proposed classification approach again several state-of-the-art algorithms using different real-world datasets and for several performance metrics: precision, recall, F1-measure, running time, Area Under the Curve (AUC) and Cohen Kappa's metric.

The rest of this paper is organized as follows. In Section 2, we review some of the most important works found in the literature concerning the resolution of online imbalanced DC traffic classification. Next, we present the mathematical concepts explored in this work and the methodology we employed in Section3. In Section 4, we give a comprehensive analysis of the results obtained for the validation of the proposed approach. Finally, we present our conclusion and anticipate future works.

II. RELATED WORK

Mice and elephant flows share resources in datacenters in a competitive way. Between the large bandwidth requirements of elephant flows and the short delays imposed by mice flows, existing traffic management strategies have to face severe resource (link and buffer) congestion and network performance degradation. In order to address this problem, existing traffic control strategies are encouraging the integration of traffic classification schemes. In this context, some of the

proposed solutions target only elephant flows because they are much more resource-hungry than mice flows [2], [11], others both elephant and mice [3], [8]. Either way, the classification algorithm has to be online and thus represent the right candidate for real-time networks. This is very challenging, considering the characteristics of network exchanges. On the other hand, data imbalance [7] must be seriously considered given the impact a non-efficient classification approach could have on traffic control strategies between the high CAPEX and OPEX costs resulting from over provisioning along with congestion and resource underutilization.

Among the different elephant flow classification approaches, the most commonly used is sampling. Sampling does not require statistics for all flows, instead is provides network measurements by pulling some packets for which classification features are measured. A well-known sampling standard is sFlow, which pulls one packet out of k entering the switch. In [2] and [11], the authors estimate pulling periods in order to reduce classification overhead. Results show that high classification performances require a large number of packets over the transmitting period, which is not adequate for early identification.

For both mice and elephant flow classification, several works have exploited inter flow correlation that took different definitions. In [8], authors present inter flow correlation as a new entity named Expanding Vector (EV) composed of 7 relationships between one entering flow and its neighbors. While authors in [3] expressed correlation through clusters formed using a semi-supervised approach of the Gaussian Mixture Model (GMM) clustering algorithm. Both approaches resulted in satisfying classification performances. However, beyond time constraints in the former and mice flow classification inaccuracies in the latter, one common limitation of the previously presented works is their lack of a mechanism to tackle the data imbalance problem.

Existing solutions to address the problem of imbalanced traffic classification fall into three categories.

- At the data level [12], [13], three approaches can be used: Under-sampling rebalances the ratio between the number of samples in each class by randomly extracting as many majority samples as possible (possibly removing significant samples). Oversampling duplicates existing minority samples (or creates synthetic samples; e.g., the Synthetic Minority Oversampling Technique, SMOTE). The hybrid approach starts by generating a few minority samples following an oversampling method, after which under-sampling operates to discard samples from majority classes or both majority and minority classes [14].
- Algorithmic solutions [7], [9] are more complex as they require a complete understanding of the classifier's reasoning in order to be able to consider the possible options of algorithm enhancement [15]. Recently, ensemble algorithms have become one of the most deployed algorithm level strategies for data imbalance problem. Even if reported as the most robust against imbalanced datasets, ensemble-based algorithms usually start by rebalancing traffic classes [10].
- For cost-sensitive strategies, the classification decision is made based on misclassification weights incorporated into the process [16]. The difficulty arises in the construction of the cost matrix. Usually, cost-sensitive methods assume the availability of a cost matrix or estimate it since this matrix should be provided experts of the domain of application [17].

It is worth mentioning that researchers have been paying more attention to this constraint (data imbalance) not only in the context of data centers but for different network environments, for example, internet. The developed algorithms for these networks are still applicable in the context of datacenters for classification problems since the only difference is usually the dimension of the classification problem.

At the data level, in [12], authors explore TCP flow classification in optically interconnected DCs, and use undersampling for features concerning the first 30 TCP packets of each flow, to mitigate the data imbalance problem. This approach includes a hash-based classifier, representing a classification memory that reduces classification latency; however, features are measured on a minimum of 30 packets, which is still too slow for real-time classification. In [13], authors propose to handle data imbalance through data augmentation. This method aims to learn the probability distribution of the features in order to be able to generate new samples, similarly to SMOTE. Along with flow level features, this work introduces sequential features representing packet directions and which are of type time series. Long Short Term Memory algorithm (LSTM) is used for the generation of sequential features, where for the numerical features, authors applied Kernel Density Estimation KDE. However, authors do not consider the time required for the traffic augmentation and extract features from 20 packets of each flow.

At the algorithm level, [9] used SMOTE to reduce the imbalances between minority and majority classes, combined with a boosting-like strategy with the aim of enhancing Byte Classification Accuracy (BCA). Along with the misclassification rate, this work computes for each sample a penalty term expressing ensemble diversity (low disagreement degree between classifiers). While the proposed algorithm exhibits excellent results for late classification, it fails to maintain its stability when used for early identification. Authors in [7] propose an extensive comparative study between several data level, ensemble algorithms and cost-sensitive approaches along with two novel ensemble algorithms that are the combination of a boosting algorithm with a strategy to deal with data imbalance (Tomek Links: TL and ROS: Random Over Sampling): TL-boost and ROSboost. The experiments have led to several results. Regarding resampling strategies, undersampling is more efficient against data imbalance as opposed to oversampling and hybrid strategies. Some ensemble algorithms were found to be more efficient, especially when combined with undersampling strategies, particularly TL-boost. As for the cost-sensitive algorithm, the authors

noted the importance and difficulty in producing an efficient cost matrix.

At the cost-sensitive level, authors in [16] propose an online classification strategy based on stream mining, for mice and flow identification, in an SDN environment. The proposed algorithm is deployed in 2 levels: 1. MetaCost, in the first classification phase at the commodity switches. It is a decision tree based cost-sensitive classification algorithm. 2. Hoeffding tree-based stream mining algorithm at the second phase, at the controller. It performs incremental learning on the first five packets of each flow, which guarantees short classification delay; although this work covers both data imbalance and online classification problems unlike most of the existing network traffic classification approach, we can cite a few limitations. Phase two classification depends completely on the results of phase one that uses only the destination port number and the protocol as training features, which may bias classification performances. Also, the paper does not discuss the impact of phase one classification on the real-time aspect of the classification.

Moreover, imbalanced data classification has been widely covered by several existing works in other fields, and the deployed algorithms can be applicable for traffic classification; however, it needs to be adapted to the statistical characteristics of traffic. LIUBoost in [18], an enhancement of RUSBoost in [19], focuses on the phenomena that cause and worsen data imbalance. Along with under-sampling, the proposed approach incorporates instances' local characteristics information in the weight calculation instead of introducing misclassification penalties, as most solutions do. Authors in [1] propose a method combining ensemble bagging with threshold moving, operating at the output of the learned model. Compared to methods using misclassification weights, this approach aims to find, for the posterior probability of different classes (obtained at the output of the bagging ensemble algorithm), the threshold that maximizes classification performances such as micro accuracy. Although results showed classification performance enhancement compared to several existing imbalanced data identification methods, these papers do not cover time constraints.

Authors in [20] challenge ensemble algorithms with another type of data: drifting data. The difficulty comes from the fact that the statistical characteristics of this type of dataset constantly change. In the proposed approach, the ensemble is periodically trained with new samples in order to increase its generalization ability, and Kappa statistics are used to tune the final voting strategy. Also, new base learners replace existing ones in order to enhance classification performances. Authors tested the efficiency of the proposed algorithm with several datasets and proved it maintains satisfying performances and overcomes data imbalance and stream drift with low resource and time consumption.

In other works, authors focused on another classification strategy that is gaining ground and is proving its efficiency in many domains, and it is trajectory clustering [21], [22]. Whether it is for face recognition, behavioral analysis, or traffic monitoring, one critical step is the extraction of the features from trajectories with different characteristics, especially when data is imbalanced [23]–[25]. We cite [23], in which authors aim to predict the risk of a driver having an accident considering his driving behavior. The feature set includes behavior features extracted from trajectories of the vehicle in movement and evaluated through XGBoost, and risk levels obtained by training a clustering algorithm on risk indicator features and also evaluated along with the resulting behavior features through Recursive Feature Elimination (RFE). This strategy is combined with undersampling and XGBoost to overcome data imbalance. Authors reported satisfying results and an important contribution regarding feature extraction for imbalanced trajectory classification problems.

Several of the presented articles report data imbalance and the importance of online network traffic classification; however, very few algorithms only can actually resolve both problems efficiently. Also, we notice that most works chose to target data or algorithm level strategies, mainly ensemble combinations with sampling because of the hardness of building a correct cost matrix, while this later could provide more accurate results [7]. Also, we believe that focusing only on the classification aspect is not sufficient to overcome these constraints and that building an approach that integrates information about traffic correlation could enhance classification performances by better characterizing data imbalance. Our approach integrates these important factors in order to deploy a novel DC traffic classification approach based on inter flow correlations. More specifically: We propose a new online traffic classification approach solving the data imbalance constraint through a cost-sensitive based strategy exploiting inter flow correlation in order to replace misclassification costs. The ensemble algorithm constructed following a bagging ensemble approach is trained using correlations obtained through a reverse k nearest neighbor algorithm on real work datasets.

III. THEORETICAL FOUNDATIONS

A. BAGGING ENSEMBLE

Ensemble algorithms are constructed from base estimators and aim to efficiently improve the generalization capability of classification algorithms [5], [15]. Statistically speaking, generalization can be expressed in terms of *bias* measuring the classifier's sensitivity toward training sets or *variance* expressing sensitivity to testing sets. Ensemble algorithms usually aim to reduce variance and thus minimize over-fitting and increase the model's robustness [15]. The most used ensemble approaches are bagging and boosting [1], [15].

- Bagging is the acronym for Bootstrapping Aggregation. Bootstrapping consists of randomly sampling the initial dataset to construct several other populations that may or may not be of the same size as the original dataset. These populations are called bags. Models are trained on these bags, and the final decision is constructed by

aggregating, following a specific approach (such as voting, weighting), the results of the learners.
- Boosting, on the other hand, sequentially trains a new model according to the results of the previous one. At each round, the new model focuses on the data samples that have been misclassified in models created in previous rounds. While the main motivation behind this approach is to create stronger learners out of weak ones, misclassifications can occur at low-level learners (at the first classification rounds) and cause performance degradation of the final algorithm.

In this work, we decided to deploy bags of Random Forests (RF) [26], in other words, to combine RF with bagging ensemble, as they both have been reported to deal successfully with variance reduction [15]. Moreover, RF is considered one of the most appropriate algorithms for online traffic classification [5] and bagging ensemble for imbalanced traffic identification [1].

B. CONCEPTUAL FOUNDATIONS OF THE RkNN ALGORITHM

Due to its numerous applications, RkNN has become an essential spatial database processing operator. It finds all points in a dataset for which a query point is part of their k-nearest neighbors set. Mainly RNN finds the points in the dataset for which the query point is their nearest neighbor [27]. If the query has the same type as the dataset, RkNN is monochromatic; otherwise, it is bichromatic RkNN [28]. In this work, we focus on monochromatic RkNN [28] since we seek to find RkNNs for testing flows among bags of training flows.

In detail, we denote the multidimensional dataset $S = \{s\}$, where s is a data object represented by m features; finding the RkNNs of a query point $q \notin S$ retrieves all points $s \in S$ for which this condition holds:

$$RkNN(q) = \{s \in S | d(s, q) < d(s, s')\}, \quad (1)$$

where s' is one of the farthest Nearest Neighbor (NN) of s and $d(a, b)$ is a distance metric (e.g.Euclidean [29]). It is important to mention that RNN and NN relationship is not symmetric. In other words, $s \in \{kNN(q)\}$ does not automatically imply $s \in \{RkNN(q)\}$, where $\{RkNN(q)\}$ represents the k-reverse nearest neighbors of q and $\{kNN(q)\}$ its k-nearest neighbors [27], [29].

Solving the RkNN problem can be done by searching for the points for which the query q is one of their nearest neighbors [27]. It determines the nearest neighbors of each point in the dataset and then goes through each one to find q, which becomes drastically time and resource-consuming with growing datasets. Another approach maps the points into a spatial index where branch and bound search is used. Using the branch and bound search, we compare the query point to each branch of a tree-like structure and remove branches that do not fit the search criteria [30].

To explain RkNN clearly, we set a fixed value of k (e.g.k=1). As mentioned earlier, the first step is to compute the NNs for every data point $s \in S$. A circle of a radius calculated as the distance between s and NN(s) is drawn, and its minimum bounding rectangle (MBR) is represented. MBRs are indexed by an R-tree according to distances between samples, but a leaf is a data point while a branch (intermediate node) is the MBR of its child node. Once we have built the tree, the RNN of a query can be obtained by measuring the distance between the MBR of each branch and the query and browsing through the different branches of the tree [29].

For our algorithm, we use another famous indexing structure, K-d tree [31], which is similar to R-tree [31] but easier to deploy and more efficient in terms of building time. The only difference between R- and K-trees is that while the former uses MBRs to group data points, the latter recursively calculates median points and splits the points into two halves around them, thus reducing the building time.

IV. PROPOSED APPROACH

Our algorithm is a modified version of the reverse kNN; we choose this ML algorithm because kNN is known to be efficient against data imbalance [32]. As noted in section two, earlier algorithms could not succeed in solving the online imbalanced traffic classification problem, which is very challenging, considering its impact on online flow scheduling. In the present section, we present our approach for imbalanced data center traffic classification combining the robustness of bagging ensemble algorithms with the power of using inter-flow correlations. This approach exploits the relationship between testing samples and flows in each training bag to act as a rebalancing weight for data center traffic mostly occupied by mice flows. The correlation measure is calculated following a Reverse k-Nearest Neighbor (RkNN) machine learning algorithm because it is able to measure a flow-to-group relationship as opposed to other algorithms like kNN for example, which is a clustering algorithm.

A. SYSTEM DESCRIPTION

Using a cost-sensitive classification approach to solve the data imbalance problem requires a cost matrix in order to encode misclassifications and construct the misclassification weights matrix. In our work, the costs in the cost matrix are represented by the correlations between testing samples and the ensemble model (inter-flow correlations). Our framework includes several modules presented in Fig. 1.

- Preprocessing module: In this work, we mainly focus on TCP flows. UDP packets that do not have a significant contribution to the total traffic are filtered out by this module. This filter can also filter useless packets (e.g., empty packets).
- Feature extraction: As reported in [33], the best classification performances were observed with flows of 8 packets. So we start by building flows from packets with the same source IP and destination IP addresses as well as port numbers, then we construct the feature matrix from 26 features (packet level and flow level features) including inter-arrival, packet length, inter-length

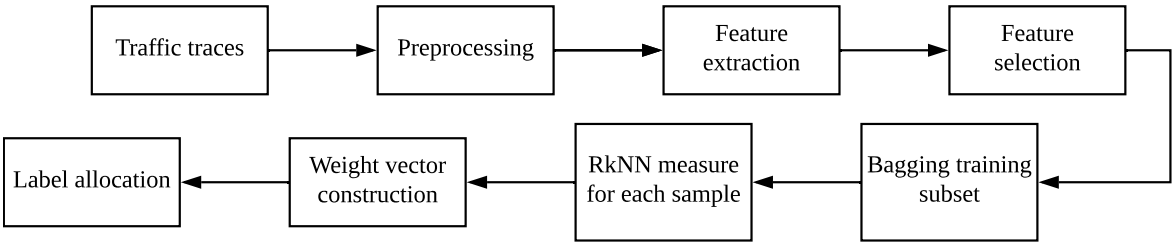

FIGURE 1. System model.

statistics, and other TCP synchronization information, such as the number of acknowledgments, push packets, synchronization packets, retransmissions, and selective acknowledgments, as in [33].

- Feature selection: We construct our final feature matrix after applying RF for feature selection. Previous research reported the limitations of the RkNN method in a highly dimensional space [27], [29], [34]. So, this step is very critical because it reduces the space dimensionality for RkNN candidates generation. Also, beyond reducing the training time, discarding useless features saves on limited memory resources.
- Bagging training samples: In this step, we build our ensemble algorithm. We start by randomly sampling the training subset with replacement to construct several other populations, namely bags (typically, these bags may or may not be of the same size as the training set). Later, we will test the performance of our algorithm using several bag numbers in order to capture the variability of the dataset and to be able to map imbalanced data into correlations efficiently. The final classification decision is made following a weighted approach to the results of the learners, where weights are interflow correlations.
- RkNN measure: During this step, we measure the correlation between each testing sample and the constructed bags. This correlation helps answer the following question: *h*ow *c*lose *i*s *t*his *f*low *t*o *t*he *p*rediction *t*hat*a* *b*ag *m*ade *?* Correlations are, in fact, the number of RkNN of a testing sample.
- Construction of weights vector: This represents the result of the RkNN algorithms or, in other words, the inter-flow correlations. For each testing sample X, we measure Rk the correlation with the i^{th} bag as follows:

$$W_i = \begin{cases} Rk(X,i), & if\ l(X) = 0 \\ Rk(X,i) * R, & \text{if l(X)=1}, \end{cases} \tag{2}$$

where R is the imbalance ratio calculated as the ratio between the number of majority flows by minority flows (in case of rare labels, the correlation is augmented by the imbalance ratio measuring the variability between the majority and minority classes), and l the classification label obtained with the bag i. Finally, we build the classification weight vector:

$$W = \{w_i\}, \quad \forall i \in [1, 2, \ldots, m], \tag{3}$$

where m is the number of bags.

- Label allocation: Depending on the predicted label for each bag and on the distance between the testing sample and the bag (correlations), we choose the label w_i associated with the maximum summation of RkNNs, as follows.

$$w_i = \begin{cases} 0, & if \sum_i w_{i|l=0} > \sum_i w_{i|l=1} \\ 1, & otherwise, \end{cases} \tag{4}$$

where $\sum_i w_{i|l=0}$ represents the sum of the number of RkNNs between the testing sample X and all the classification bags for which the predicted label is equal to 0 and $\sum_i w_{i|l=1}$ is the sum of the number of RkNNs between the testing sample X and all the bags for which the predicted label is equal to 1. The final algorithm is summarized in Fig.2.

B. REVERSE K-NEAREST NEIGHBOR PROPOSED ALGORITHM FOR CORRELATION COMPUTATION

In this section, we present the approach to compute the number of RkNNs of each testing flow. The authors of [27] claimed that when using Range Query techniques, MBR based, searching for the specific data points or even their number in the range can consume many resources and much time. However, since our approach is based on a radius query instead, distance calculations are faster and less computationally costly. Also, the dimensionality of the feature vector needs to be considered because the higher dimensionality results in lower pruning efficiency. This is caused mainly by the high overlap between circles, which can impel the algorithm not to prune them. Our solution to this constraint is to use feature selection, as it represents an efficient way to reduce dataset dimensionality by removing only nonsignificant features.

Algorithm 1 describes the proposed approach. We measure the correlation between incoming flows and the training model on several bags in two sub-functions (representing both the training and testing strategies of the machine learning algorithm). In the first part of our algorithm: training

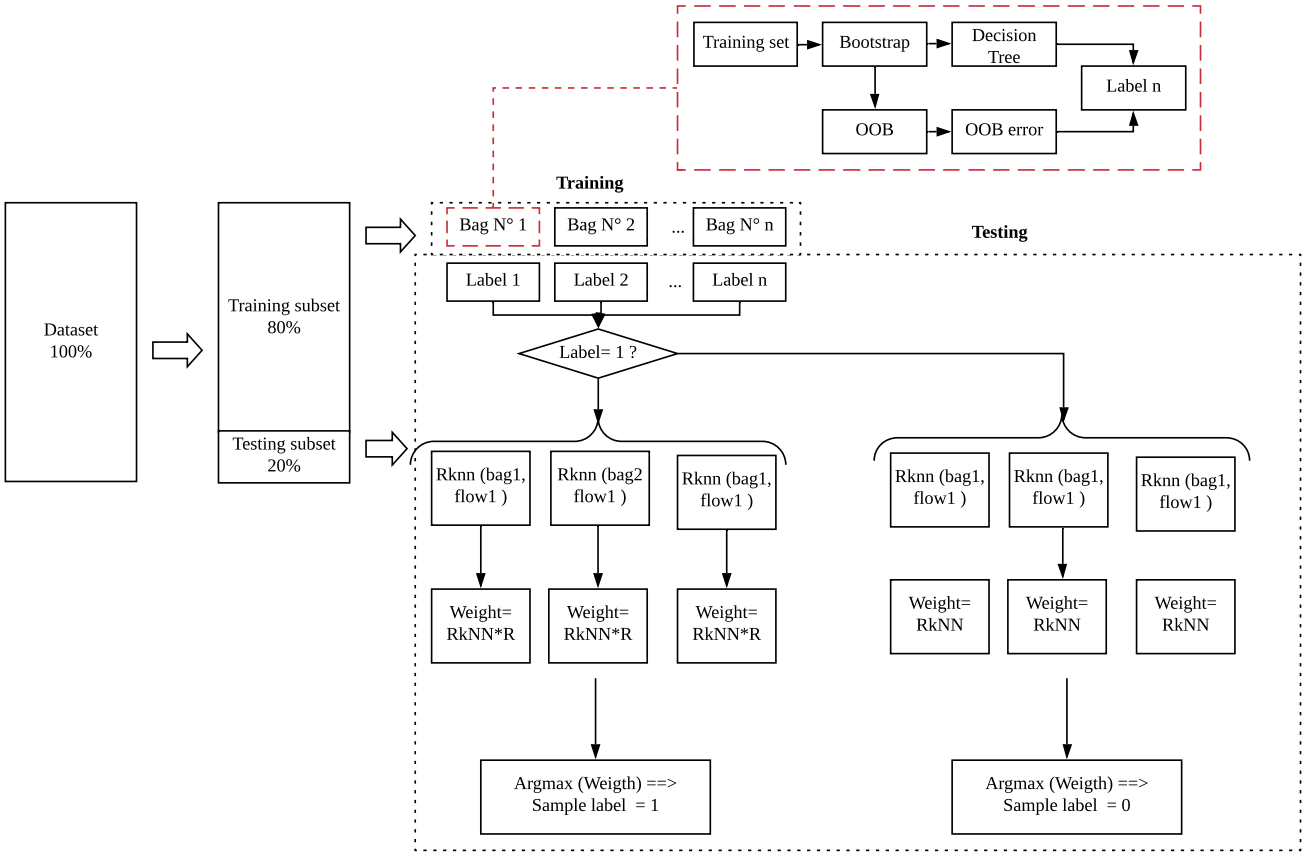

FIGURE 2. Proposed approach.

(steps 3-11), we construct a K-d tree using training samples of each bag s_{ij}. During this step, we start by computing for each data point s_{ij} (from bag i), its kNNs in order to later concentrate the search of RkNNs into the subset of k data points closest to s_{ij}. This filtering operation (steps 5-8) is followed by the construction of the index structure (steps 9-10). We draw circles with the radius corresponding to the distance between the sample and its farthest neighbor in order to include all the calculated kNNs. After this, we group circles two by two until we reach the tree root; this depends on the distance between the centers of the circles. In other words, we compute the nearest neighbor of a circle's center, and if this nearest neighbor's circle has not already been grouped to another circle, it is chosen. Otherwise, we choose the next nearest neighbor.

For testing, considering a set $Q = \{q\}$ of queries representing testing samples (flows) in a multidimensional space (features), following the second sub-function of our algorithm (steps 13 to 27), we browse through the constructed K-d tree (from root to leaves), in order to construct RkNN, a vector containing the number of RkNNs of each sample. The procedure begins by measuring the distance between the query and the center of each circle (representing a branch or a leaf in the tree). We compare this distance to the radius of the circle in order to choose the branches respecting the condition in step 15. Once we arrive at the bottom of the tree, we increment the RkNN number of the concerned sample by 1 (steps 17-18). Otherwise, we continue browsing through the tree (steps 20 and 21).

V. EXPERIMENTS

A. DATASETS

We conducted our experiments on a cloud data center dataset (CAIDA [35]), two university datacenters datasets (Wisconsin Madison, UNI1 and UNI2 datasets [36])) and an internet traffic dataset (UNIBs [37]).

CAIDA contains a series of datasets captured in different directions uplink (direction A) or downlink (direction B) at different periods; We used traffic traces produced February 17, 2011, and in direction A. We decided to use traffic traces from CAIDA's 2011 data set because it has been recently used in a real-time traffic prediction paper [38]. Also, we choose direction A instead of direction B since the later contains a significant amount of packet loss according to [35]. The datasets are highly imbalanced [35], [36], and contain unknown TCP and UDP packets, making them more suitable for mice/elephant classification rather than application identification. As for UNI1 and UNI2 [39], the traffic traces were collected from one university site in the western US for

Algorithm 1 RkNN Based Approach for Correlation Measure: CCS

Input:

$$k, Q = \{q\}, all = \{S_i\}, S_i = \{s_{i,1}, s_{i,2}, \ldots, s_{i,j}\}$$

Output: R_k

▷ Training : Build K-d Tree for each bag

repeat
  foreach *sample s in S* **do**
    Compute $R_k\{s\}$
    Draw a circle with the radius equal to the distance with the farthest NN
  end
  Group circles 2 by 2 in order to construct the branches of the tree.
  Repeat until arriving to the root
until *There is no bag in all*;
▷ Testing : Calculate RkNN for each testing sample for each bag
repeat
  foreach *circle (C) composing the K-d tree* **do**
    if *d(q,center(C: a branch or the leaf))< radius C* **then**
      if *C is leaf* **then**
        $R_k = R_k + 1$
      end
      else
        Go to children of this branch
        Go to 15
      end
    end
  end
  Return R_k
until *There is no bag in all and this for each query q in Q*;

12 hours over several days. After filtering out nonsignificant packets, we combined several traffic traces files in chronological order to construct a file with enough instances. UNIBs, on the other hand, is a traffic dataset composed of 99% TCP packets exchanged between several machines connected to the internet through an edge router and a high capacity link.

Caida's dataset forms a (20000 × 26) feature matrix, UNI1's dataset a (21013 × 26) matrix, UNI2's dataset a (2586 × 26) matrix and UNIBs dataset a (19349 × 26) matrix, where the rows represent traffic packets grouped into flows by 4 parameters: source and destination addresses, and port numbers. The protocol type is not included since we applied a filter at the preprocessing step to select only TCP packets) and the columns represent the feature vector as described in [33].

It is important to mention that since we were unable to find a ground truth labeled using elephant and mice flow tags, we constructed our own ground truth by putting this assumption: If the number of packets in the flow exceeds 100 packets, it is an elephant flow; otherwise, it is a mice flow. Our assumption is based on the fact that if Ethernet frame size is limited to 1500 bytes (jumbo frames excluded), 100 frames generate 1.2 Mbits of data. We assume this size could represent an elephant flow [39]. However, as mentioned in [39] for UNI2 dataset and as observed from our experiments for UNIBs dataset, flows are short and thus hold a smaller number of packets. Thereby, according to observations made from [39] and from experiments, we fixed the threshold to 15 packets for UNI2 and 30 packets for UNIBs.

B. PERFORMANCE EVALUATION

In the present section, we test the performances of our classifier by varying its hyperparameters: the number and size of bags (3, 5 and 10 of 20, 40, 60 and 80% of the training set) and the RkNN order (k=2, 5, 10 and 20). We also compare the best classification performances we obtain with our correlation-based algorithm, to 16 existing algorithms for imbalanced traffic classification ranging from undersampling, oversampling, and hybrid approaches, algorithm level strategies to cost sensitive different algorithms.

1) PERFORMANCE METRICS

While we usually use accuracy to test the performance of a classifier, with a highly imbalanced dataset, this measure biases any algorithm's efficiency, since it focuses mostly on majority classes. Consequently, in order to efficiently validate our proposed approach, we used several performance metrics: precision, recall, F1-measure, the confusion matrix, and running time. Although the F1 measure (combining precision and recall) is the most commonly used metric when dealing with imbalanced classes, we seek to find a good balance among the five metrics. This is because, as will be shown in the different scenarios covered for this performance testing, in some cases, we may find a higher F1 measure, which is mainly obtained from higher precision or higher recall, and not a balance between the two. It is important to mention that the graphs we present are related to the minority class (elephant flows) and that the majority class achieves between 98% and 99% precision, recall, and F1 measure in most scenarios. Since our algorithm is as efficient as the others for the majority class, we decided to focus on the rare samples in this analysis. F1 measure is calculated using the following equation:

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}. \quad (5)$$

Precision expresses the relevance of the results, for example, out of 100 samples, how many were correctly classified as class 1 or 2 (True Positives, TP) among the total number of samples predicted as positive (True Positives + False Positives, TP+FP), which is equivalent to the following equation:

$$Precision = \frac{TP}{TP + FP}. \quad (6)$$

Recall, on the other hand, expresses the probability of detection, in other words, the ratio between the positive

predictions (TP) and the total positive samples (True Positives + False Negatives, TP + FN),

$$Recall = \frac{TP}{TP + FN}. \quad (7)$$

High precision and low recall are equivalent to high FN and low FP (a significant number of mice flows are classified as elephant flows), while high recall and small precision mean high FP and low FN (a significant number of elephant flows are classified as mice). Following our strategy of balancing between precision and recall, we try to avoid congestion and overprovisioning at the same time, while maintaining an efficient training time for time constraints.

When comparing the optimal results obtained following our approach to other related works, we use two other metrics along with precision, recall, F1 measure and running time; it is the Area Under the Receiver Operating Characteristic (ROC) Curve: AUC and Cohen's Kappa.

AUC metric is obtained from the ROC curve that plots the True Positive Rate against False Positive Rate in order to represent the model's performances. TPR and FPR are calculated through the evaluation of a logistic regression model for different thresholds. Because it does not require any information about the traffic distributions, AUC is highly used for performance evaluation in presence of data imbalance [40]. AUC calculates the area under the curve between TPR and FPR values both equal to 0 and 1 (0, 0) and (1, 1) using integral calculus. The higher is the value of AUC; the better is the class's separability of the algorithm [41].

As for Cohen's Kappa metric, it measures the agreement between two algorithms also called raters, when applied to the same dataset. The objective is to measure inter-rater reliability observed with higher values of Kappa. Usually, Kappa values vary between 0 and 1, where 0 represents no agreements between the classifiers or an agreement by chance, and 1 is the synonym of an ideal agreement. In order to calculate kappa (k), we construct a matrix including the probabilities that both algorithms obtain the same labels (P_{11}, P_{22}) and the probability that they don't agree about the label assignation (P_{12}, P_{21}). $\left[\begin{smallmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{smallmatrix}\right]$, We then measure $_{P0}$ from P_{11} and P_{22} expressing the level of agreement and Pe from P_{12} and P_{21} expressing random label allocation from the algorithms. Finally, Kappa value is obtained from P_0 and P_e following this equation [42].

$$\kappa = \frac{P_0 - P_e}{1 - P_e} \quad (8)$$

2) EXPERIMENTAL RESULTS AND ANALYSIS

Our algorithm is based on the computation of correlations between each testing flow and the training samples composing each bag of the bagging ensemble. Thus, it is important to tune the classifier to the best bag size and number. On this basis, we established a comparison between several scenarios for 3, 5, 10, and 20 bags using respectively 20%, 40%, 60%, and 80% of the training set in each bag. Among all the tested scenarios we present in this section, we choose for each data set, the one with the best classification performance in order to compare it with other imbalanced classification algorithms. For the first data set (CAIDA), we will thoroughly analyze the results and present the approach that leads to choosing one scenario, among others. For the rest of the datasets, because of limited space, we will just highlight the optimal performances. Because the running time for scenarios deploying 20 bags of any size and for any dataset is very high and thus not adequate for real-time constraints, we decided not to include their results in the following experimental result analysis.

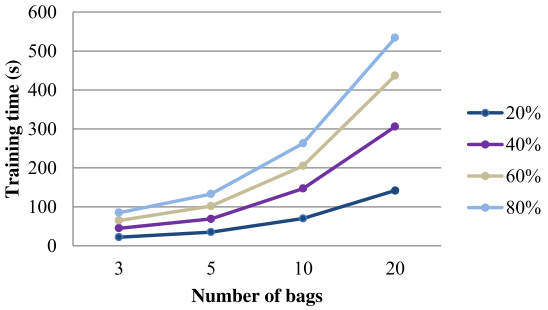

FIGURE 3. Training time evolution according to the number and size of the bags(CAIDA).

- Choosing the number and size of bags:
  For CAIDA's dataset, we notice that except for the scenario deploying 10 bags of 20% of the training set, increasing the number of bags and their density enhances classification performances in terms of precision, recall, F1 measure and AUC. However, since training and testing times grow exponentially according to the number of bags(Fig.3 for training time), we need to find the best compromise considering the confusion matrix in order to select the optimal scenario. We select each time the best scenario among the four representing the same number of bags and various sizes, then we compare these 4 scenarios (one with the number of bags=3, another with 5 and then 10) and select the best compromise in terms of classification metrics (precision, recall, F1-measure, along with the confusion matrix and AUC) and time consumption (running times). In the first step, we select (from Fig.5) the scenario deploying 20% of the training set in each of the 3 bags corresponding to 65% precision, 61% recall, 63% F1 measure, 80% AUC and confusion matrix $\left[\begin{smallmatrix} 3807 & 48 \\ 56 & 89 \end{smallmatrix}\right]$ for 22 seconds of training, as the best scenario with 3 training bags. With 5 bags, the optimal scenario (65% precision, 61% recall, 63% F1 measure, 80% AUC and confusion matrix $\left[\begin{smallmatrix} 3808 & 47 \\ 56 & 89 \end{smallmatrix}\right]$ balancing between FP and FN) is obtained with 20% of the training set in each bag, for 36 seconds of training. Next, with 59% precision, 72 % recall, 65% F1 measure, 85% AUC and confusion matrix $\left[\begin{smallmatrix} 3781 & 74 \\ 40 & 105 \end{smallmatrix}\right]$, the best tradeoff is obtained with 40% training set in each of the 10 bags in 147 seconds of training. Finally, for 22 seconds of training time, the scenario using three bags of 20% of the

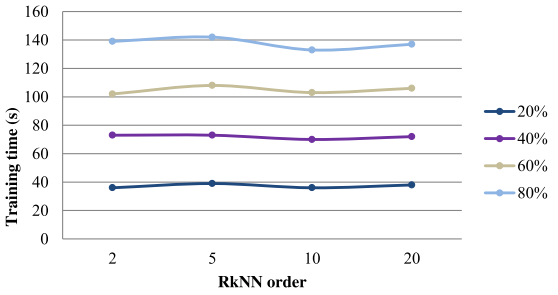

FIGURE 4. Training time evolution according to RkNN order (CAIDA).

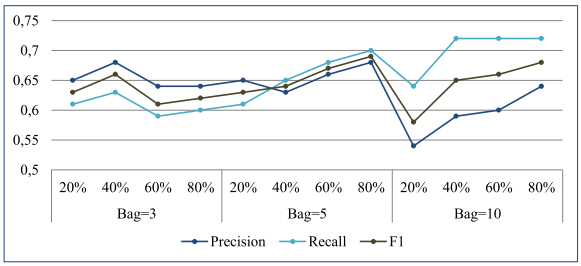

FIGURE 5. Classification performance of the proposed approach using different bag numbers and sizes (CAIDA).

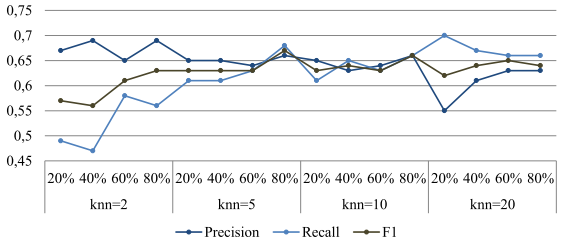

FIGURE 6. Classification performance of the proposed approach with 5 bags for different RkNN order (CAIDA).

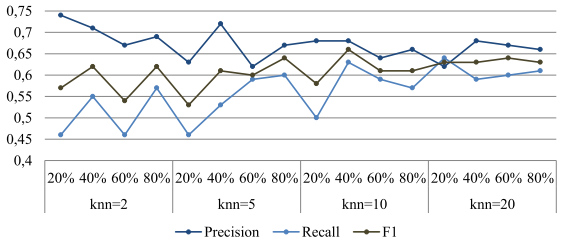

FIGURE 7. Classification performance of the proposed approach with 3 bags for different RkNN order (CAIDA).

original training set represents the optimal compromise with an RkNN order equal to 10.

For the second dataset (UNI1), when searching for a scenario with balanced confusion matrix (number of FN close to FP) in order to avoid over provisioning and congestion both degrading QoS performances, and while considering the compromise with precision, recall F1 measure and AUC, we find that the best scenario is obtained with 5 bags of 40% of the training set. Results include 55% precision, 58% recall, 57% F1 measure, almost 80% AUC, a confusion matrix $\left[\begin{smallmatrix}3847 & 114\\ 101 & 140\end{smallmatrix}\right]$ for 99 seconds training time.

As opposed to the other datasets, the best scenario for UNI2 is obtained within 3.24 seconds with 3 bags of 60% of the training set in each bag. 82% precision, 100% recall, 90% F1 measure, 99% AUC and a confusion matrix $\left[\begin{smallmatrix}478 & 7\\ 0 & 33\end{smallmatrix}\right]$ result from this dataset. It is worth mentioning that this dataset is also shorter than the others.

For UNIBS dataset, we notice from our experiments that it is also characterized by small flows, we tune the threshold and fix it to 30 packets per-flow and obtain the optimal results with 5 bags of 20% of the training set. Classification performances include 65% precision, 78% recall, 71% F1 measure, almost 80%AUC, a confusion matrix equal to $\left[\begin{smallmatrix}2119 & 521\\ 270 & 960\end{smallmatrix}\right]$ in 54 seconds. Other scenarios present better classification performances; however, their training time is too high.

- Varying the RkNN order:

For the next experiments, we focus on enhancing the performance of our classifier by tuning the order of the RkNN algorithm. In other words, since the RkNN computes the number of bag samples that have a testing sample (query) among their k-nearest neighbor set, we want to capture how the number of nearest neighbors "k" we set for each bag, affects the classification performance. Before going through the scenarios, it is worth noticing in Fig.4 that the RkNN order only slightly (almost does not) affects the training time, since it is involved in the generation of the circles (choose the k farthest NN to determine the radius), which affects the performance of the classification in terms of precision, recall, F1 measure, and confusion matrix but not the training and testing times. As opposed to the number and size of the bags, affecting the number of circles involved in the training process and thus in the training periods, as shown in Fig.3.

Since scenarios using 3 and 5 bags manage to find good tradeoffs between precision, recall, F1 measure, and confusion matrix, as well as training and testing times compared to scenarios with 10, we decided to limit our tests for varying the RkNN to them.

For CAIDA dataset, first, it is worth observing in Fig. 6 and 7 that increasing the RkNN order increases the stability of the classification. We follow the same approach as in the previous section, to select the optimal scenario. With 5 bags (as shown in Fig.6), the best compromise is obtained with an RkNN order equal to 10, when using 20% of the training set in each bag. This scenario provides 65% precision, 61% recall, 63% F1 measure, a confusion matrix equal to $\left[\begin{smallmatrix}3808 & 47\\ 56 & 89\end{smallmatrix}\right]$ in 36 seconds of training. With 3 bags (Fig.7), although a scenario with an RkNN order set to 10 for 40 % training set in each bag outperforms all the scenarios with 68% precision, 63% recall, 66% F1 measure and

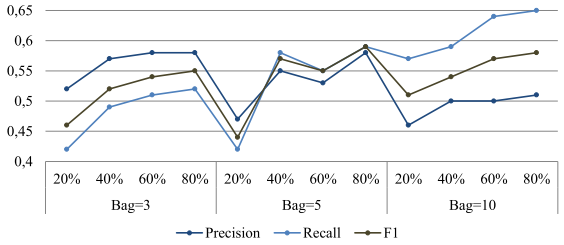

FIGURE 8. Performance of the proposed approach using different bag numbers and sizes (UNI1).

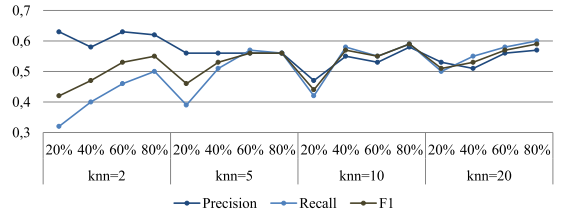

FIGURE 9. Performance of the proposed approach with 5 bags for different RkNN order (UNI1).

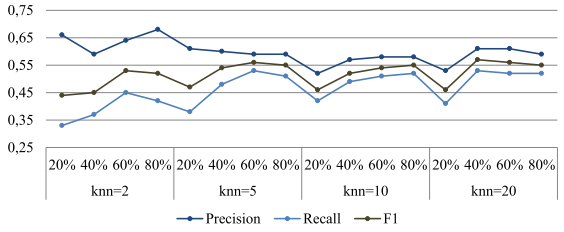

FIGURE 10. Performance of the proposed approach with 3 bags for different RkNN order (UNI1).

$\begin{bmatrix} 3812 & 43 \\ 53 & 92 \end{bmatrix}$ confusion matrix, these results are obtained in 42 seconds while the scenario using RkNN order equal to 20 with bag sizes of 20% of the training data set provides 62% precision, 64% recall, 63% F1 measure and $\begin{bmatrix} 3798 & 57 \\ 52 & 93 \end{bmatrix}$ confusion matrix in 22 seconds of training time only. Finally, we consider the last scenario as optimal for the CAIDA dataset.

For the second dataset (UNI1, in Fig.8, 9 and 10), with 5 bags and kNN=10, the best scenario is still 55% precision, 58% recall, 57% F1 measure, 80% AUC and a well-balanced confusion matrix, it is obtained with bags of 40% of the training set. This scenario is outperformed by another one with 3 bags of 40% and kNN=20, resulting in 61% precision, 53% recall, 57% F1 measure, 76% AUC and a confusion matrix $\begin{bmatrix} 3877 & 84 \\ 113 & 129 \end{bmatrix}$ in 63 seconds.

For UNI2 dataset (in Fig.11, 12 and 13), the optimal scenario remains the same, in other words, 3 bags of 60% with kNN=10, resulting in 82% precision, 100% recall, 90% F1 measure, 99% AUC and a balanced confusion matrix for 3.24 seconds.

For UNIBs (in Fig.14, 15 and 16), we enhance the optimal scenario found represented by 5 bags of 20% and kNN=10, by the scenario with kNN=5 obtaining 69%

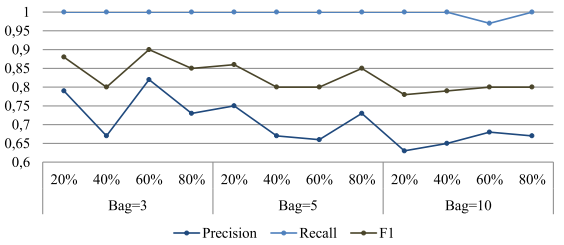

FIGURE 11. Performance of the proposed approach using different bag numbers and sizes (UNI2).

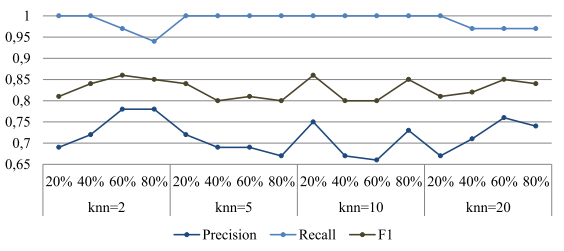

FIGURE 12. Performance of the proposed approach with 5 bags for different RkNN order (UNI2).

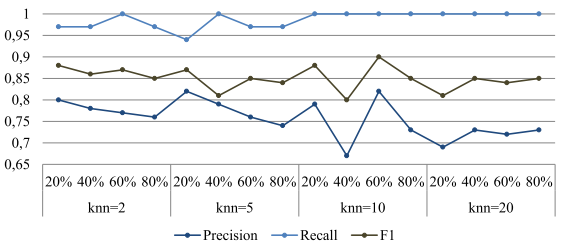

FIGURE 13. Performance of the proposed approach with 3 bags for different RkNN order (UNI2).

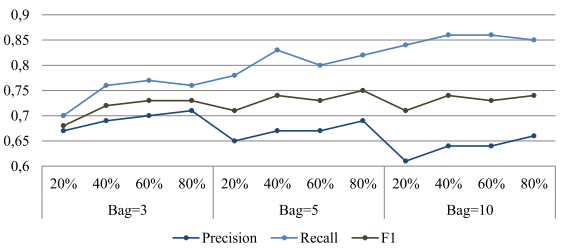

FIGURE 14. Performance of the proposed approach using different bag numbers and sizes (UNIBs).

precision, 71% recall, 70% F1 measure, 78% AUC and a confusion matrix equal to $\begin{bmatrix} 2291 & 349 \\ 403 & 827 \end{bmatrix}$ for 53.31 seconds. This scenario can be even more enhanced specially in terms of confusion matrix using 3 bags of 40% and a kNN order equal to 5. The optimal scenario for UNIBs dataset obtains 71% precision, 73% recall, 72% F1 measure, 80% AUC and a confusion matrix equal to $\begin{bmatrix} 2272 & 368 \\ 335 & 895 \end{bmatrix}$ in 68.31 seconds of training.

- Performance comparison:
  Existing classification strategies that handle imbalanced data fall into three categories: data level

TABLE 1. Performance metric comparison between our approach and existing ones for CAIDA and UNI1 datasets.

Dataset	CAIDA						UNI1					
Algorithm	Precision	Recall	F1	AUC	Time(s)	Kappa	Precision	Recall	F1	AUC	Time(s)	Kappa
SMOTE	0,51	0,74	0,61	0,85	4,36	0,663	0,48	0,71	0,58	0,83	4,25	0,62
kMeans Smote	0,75	0,45	0,56	0,72	3	0,63	0,67	0,44	0,53	0,71	4,55	0,65
ROS	0,67	0,58	0,62	0,78	2,19	0,7	0,61	0,58	0,59	0,77	2,57	0,7
ADASYN	0,51	0,73	0,6	0,85	4,75	0,7	0,47	0,69	0,56	0,81	4,56	0,61
RUS	0,24	0,97	0,38	0,92	0,13	0,36	0,23	0,88	0,37	0,85	0,28	0,31
TomekLink	0,68	0,52	0,59	0,75	1,66	0,7	0,65	0,5	0,57	0,74	3,21	0,68
Condenced NN	0,61	0,57	0,59	0,779	344,77	0,619	0,52	0,62	0,56	0,79	589,88	0,60
Edited NN	0,56	0,74	0,64	0,86	1,73	0,73	0,5	0,67	0,57	0,81	2,82	0,697
Repeaded NN	0,49	0,8	0,61	0,88	5,66	0,7	0,43	0,71	0,53	0,84	9,41	0,627
All kNN	0,52	0,77	0,62	0,87	3	0,73	0,46	0,71	0,56	0,82	5	0,64
Hybrid	0,52	0,75	0,61	0,86	5,56	0,66	0,47	0,72	0,57	0,83	7,08	0,61
Balanced RF	0,27	0,96	0,42	0,929	1,32	0,4	0,3	0,88	0,44	0,87	2,34	0,41
RUS Boost	0,39	0,49	0,43	0,73	0,76	0,4	0,36	0,6	0,45	0,76	1,18	0,42
CS (J48)	0,7	0,32	0,44	0,66	0,25	0,43	0,63	0,32	0,42	0,65	0,53	0,4
CS (Bagging)	0,78	0,31	0,45	0,66	0,89	0,44	0,91	0,238	0,37	0,61	1,59	0,36
CS (RF)	0,78	0,32	0,45	0,66	3,03	0,45	0,8	0,295	0,43	0,64	5,43	0,41
Our approach	0,62	0,64	0,63	0,80	22	/	0,61	0,53	0,57	0,78	63	/

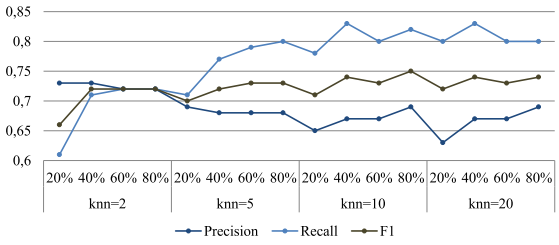

FIGURE 15. Performance of the the proposed approach with 5 bags for different RkNN order (UNIBs).

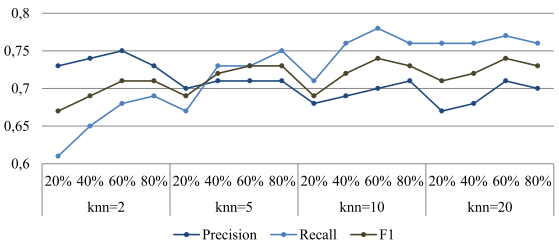

FIGURE 16. Performance of the proposed approach with 3 bags for different RkNN order (UNIBs).

algorithms, algorithm level strategies, and cost-sensitive approaches. Since most traffic classification methods rely on data level approaches or on ensemble algorithms [14], we propose to compare our performances to five undersampling algorithms (Edited Nearest neighbor, Repeated Nearest neighbor, All kNN, RUS, TomekLink), four oversampling algorithms (ADASYN, KmeansSMOTE, SMOTE, ROS), one hybrid (SMOTE with TomekLink) and two algorithm level strategies (a data level balanced Random Forest-based bagging, RUSBoost). Details about these algorithms can be found in [7]. Also, we compare in this section, our correlation-based cost-sensitive strategy for online imbalanced DC traffic classification to three other cost-sensitive algorithms with respectively J48 method, bagging strategy, and Random Forest set as base learners. It is worth mentioning that the cost-sensitive algorithm implementing bagging as a base learner is compared to MetaCost, an algorithm often cited when introducing cost-sensitive classification [7], [43]. Meta in MetaCost refers to meta-learning defined in [44] as the process of learning from various accumulative experiences. Ensemble algorithms belong then to the meta learners family. We calculated the cost matrix for cost-sensitive algorithms such as in [7], following the equation below. In other words, misclassification cost of class i as j, C_{ij} depends on the number of flows in each class M_i or M_j.

$$C_{ij} = \begin{cases} log_{10}(M_i)/log_{10}(M_j), & if\ i \neq j \\ 0, & otherwise, \end{cases} \quad (9)$$

Experiments were conducted on four different datasets, and are summarized in Table 1 and 2. Classification performance metrics used to compare the algorithms are precision, recall, F1 measure, AUC, and Kappa.

It is worth mentioning that Kappa metric is measured between the ground truth labels, the labels obtained through our approach, and the labels obtained through the algorithm we are comparing our algorithm to, reason why there is no Kappa value in the row ''our approach'' of each of the tables.

- CAIDA dataset: First, for AUC and kappa metrics, we notice that oversampling undersampling and hybrid algorithms result in good values. Ensemble algorithms result in higher AUC ranging from 73 to 92% expressing the classifier's high degree of separability between classes but in 40% Kappa, which indicates a weak agreement between classifiers. Finally, cost-sensitive algorithms exhibit acceptable AUC and a weak Kappa. Regarding Precision, Recall, and F1 measure, we notice that all algorithms result in either high precision and low recall or vice versa and never a balance between the two. Compared to these algorithms, our proposed approach

TABLE 2. Performance metric comparison between our approach and existing ones for UNI2 and UNIBs datasets.

Dataset	UNI2						UNIBs					
Algorithm	Precision	Recall	F1	AUC	Time(s)	Kappa	Precision	Recall	F1	AUC	Time(s)	Kappa
SMOTE	0,82	0,94	0,87	0,96	0,1	0,9	0.73	0.78	0.75	0.82	2,61	0,73
kMeans Smote	0,78	0,94	0,85	0,978	0,31	0,88	0.79	0.69	0.74	0.80	2,88	0,735
ROS	0,73	0,97	0,83	0,97	0,07	0,9	0.75	0.74	0.75	0.81	2,35	0,75
ADASYN	0,79	0,94	0,86	0,86	0,12	0,88	0.71	0.81	0.75	0.82	3,58	0,72
RUS	0,66	1	0,8	0,98	0,05	0,95	0.67	0.85	0.75	0.82	1,05	0,71
TomekLink	0,86	0,73	0,79	0,85	0,09	0,71	0.75	0.74	0.75	0.81	2,26	0,75
Condenced NN	0,89	0,73	0,8	0,86	1,93	0,78	0,63	0,83	0,71	0,80	911,87	0,59
Edited NN	0,77	1	0,87	0,989	0,08	0,93	0.64	0.89	0.74	0,82	1,82	0,707
Repeaded NN	0,63	1	0,75	0,98	0,124	0,93	0.60	0.91	0.73	0.81	3,62	0,656
All kNN	0,63	1	0,78	0,98	0,12	0,93	0.62	0.90	0.73	0.82	2,84	0,7
Hybrid	0,74	0,94	0,83	0,859	0,15	0,952	0.72	0.78	0.75	0.81	3,52	0,74
Balanced RF	0,65	1	0,79	0,9814	0,17	0,94	0.71	0.81	0.76	0.83	7	0,75
RUS Boost	0,76	0,76	0,76	0,87	0,19	0,92	0,77	0,62	0.9	0.76	3	0,71
CS (J48)	0,7	0,909	0,88	0,95	0,01	0,88	0,86	0,87	0,86	0.77	0,54	0,56
CS (Bagging)	0,87	0,901	0,88	0,95	0,03	0,88	0,88	0,88	0,88	0,81	1,6	0,62
CS (RF)	0,87	0,91	0,89	0,95	0,1	0,88	0,89	0,89	0,89	0.82	5,54	0,64
Our approach	0,82	1	0,9	0,99	3,24	/	0,71	0,73	0,72	0,80	68,31	/

results in a more balanced performance metrics with 62% precision, 64% recall, 63% F1 measure and a higher AUC (80%) in 22 seconds of training.

- UNI1 dataset: We notice a good agreement between our classifier and the oversampling, most undersampling (except RUS), and hybrid strategies with Kappa values around 70%, but on the other hand, smaller Kappa values with ensemble and cost-sensitive strategies which exhibits the worst classification performances. Our scenario outperforms the best classifier (TomekLink) in terms of recall and AUC.
- UNI2 dataset: We notice more stability with higher values for all classification performances for almost all algorithms, even ensemble and cost-sensitive algorithms resulting so far in weak classifications. Kappa values are approaching 90% while AUC are acceding 90%, thus excluding the possibility of a random classification using our approach. It is worth mentioning that cost-sensitive approaches provide the best performances for UNI2 as opposed to previous datasets. Although few seconds slower, our approach outperforms all algorithms and results in 82% precision, 100% recall, 90% F1 measure and 99% AUC.
- UNIBs dataset: Lastly, with UNIBs datasets we also observe good Kappa values for all algorithms (higher than 70% but a little less for cost-sensitive algorithms). Also, most AUC values exceed 80%. As for precision, recall, and F1 measure, they fluctuate around 70%, for oversampling and ensemble algorithms and have values of 60%, 70% and 70% respectively for undersampling. Cost-sensitive algorithms exhibit significant enhancements and outperform the rest of the methods. Our approach obtains satisfying results estimated at 71% precision, 73% recall, and 70% F1 measure, which is better than several algorithms, especially undersampling strategies and which also approaches the optimal ones.

It is worth noticing that for all datasets, the worst algorithm in terms of running time is Condensed Nearest neighbor. However, it produces better results than the fastest algorithm, which for most datasets is RUS. For CAIDA for example, CNN (344.77 seconds of training) provides 60% precision, 57% recall, 59% F1 measure, 78% AUC and 62% kappa as opposed to 24% precision, 97% recall, 38 % F1-measure, 92.7% AUC and 36% Kappa with RUS (0.13 seconds).

In summary, regarding our approach, although slower than most of the others, it mostly outperforms the other algorithms in regard to precision, recall, and F1 measure and exhibits a satisfying AUC and Kappa values. If we take a closer look at the obtained results, we discuss that classification performances are not directly related to the level of imbalance, for example, if we compare CAIDA and UNI1, CAIDA is more imbalanced than UNI1, however, our algorithm results in better performances (62% precision, 64% recall, 63% F1 measure, and 80% AUC for CAIDA as opposed to 61% precision, 53% recall, 57 % F1-measure, and 78% AUC for UNI1). Also, these performances are not related to the size of the dataset (although this latter parameter does affect the running time of our algorithm). These performances can be related to the quality of traffic traces.

An advantage of our solution is its ability to select any RkNN order. Indeed, higher or lower RkNN orders hardly affects the training time. Thus, we can efficiently tune the hyperparameters of our approach without increasing the running time. However, although efficient, the Random Forest algorithm can make the classification, resource-hungry, and time-consuming, depending on the number of trees. Consequently, we will test the effectiveness of our approach, with other pruning strategies and other base learners in future works. As for testing time, we will develop a strategy that will not measure the RkNN correlation for each testing sample.

VI. STATISTICAL ANALYSIS

Through the previous section, we studied the efficiency of our proposed approach against data imbalance and for online classification. We also compared obtained results following

TABLE 3. Statistical analysis, Friedman's test.

	Precision	Recall	F1-measure	AUC
Friedman's score	35.025	38.658	44.354	34.682
P_values	<0.0001	<0.0001	<0.0001	<0.0001

the best scenario for each dataset to classification performances of state of the art approaches. However, in order to validate the statistical significance of these results, it is important to establish statistical analysis between the different algorithms on the various datasets [7]. Indeed, when several classifiers are applied on diverse datasets, their performances must be different, and it is important to test whether these differences are random or real [45].

Two categories of statistical analysis methods exist parametric and nonparametric. While the former is build on several hypotheses regarding data distributions, the latter does not require any assumptions. When data normality and variance equality or such as in our case the distribution is unknown, nonparametric methods are the best choice [46]. Therefore, we have decided to statistically validate our results through Friedman's test, one of the most used nonparametric approaches [45].

Friedman's test consists of confirming or not a null hypothesis that states that repeated measurements obtained differently have the same distribution. In other words, the different algorithms have similar behavior in all datasets. We aim to reject the null hypothesis and thus prove the statistical difference between the deployed algorithms.

Friedman's test starts by sorting the algorithms according to the value of the classification performance for each dataset, after which it allocates a score to each algorithm depending on its rank. Finally, the Friedman score χ_F^2 is calculated following the bellow equation [exploratory]. Where N represents the number of datasets on which tests have been established, k the number of trained algorithms, and finally R_j is the score obtained after sorting the algorithms.

$$\chi_F^2 = \frac{12N}{k(k+1)}\left[\sum_j R_j^2 - 0.25k*(k+1)\right] \tag{10}$$

In Table 3, we present the Friedman's scores obtained when comparing oversampling, undersampling, hybrid, algorithm level, and cost sensitive approaches on different datsets. In order to analyze these values, we also measure their corresponding P-values. Similarities between algorithms are expressed through high values of p-value, on the opposite, smaller values express statistical significance. Judging from the very low values of P-value, we can confirm the statistical differences between the algorithms.

VII. CONCLUSION AND FUTURE WORKS

In this paper, we propose a novel classification approach combining the efficiency of cost-sensitive methods and the robustness of bagging ensemble approaches through an inter-flow correlation-based strategy. After training several Random Forests on bags of data traffic, we use the RkNN algorithm to measure the correlation between testing samples representing traffic flows and each bagging classifier to compute the rebalancing weights. To the best of our knowledge, no such combination has been proposed to overcome the challenges of imbalanced traffic while respecting time constraints. To evaluate our approach, we vary the hyperparameters of our algorithm and compare its performances on data center traffic traces to other approaches for imbalanced traffic classification. Our correlation-based classification approach outperforms other algorithms in regard to precision, recall, and F1 measure by rebalancing the training set following a cost-sensitive way that avoids the use of a misclassification cost matrix. Our algorithm balances the trade-off between precision and recall and produces a confusion matrix with good FP and FN ratios that prevent over-provisioning as well as congestion. By correctly tuning our algorithm, the results show running times.

For future works, we will improve the cost-sensitive approach of our RkNN-based correlation algorithm in order to enhance classification performances, particularly in terms of computational time, by substituting the tree-like strategy with an efficient clustering approach. Also, we will include UDP packets, which introduce more challenges in the initialization step of constructing the ground truth and improve the robustness of our proposed approach against larger traffic datasets.

ACKNOWLEDGMENT

The authors would like to thank Chuan Pham for his help.

REFERENCES

[1] G. Collell, D. Prelec, and K. R. Patil, "A simple plug-in bagging ensemble based on threshold-moving for classifying binary and multiclass imbalanced data," *Neurocomputing*, vol. 275, pp. 330–340, Jan. 2018.

[2] F. Tang, L. Li, L. Barolli, and C. Tang, "An efficient sampling and classification approach for flow detection in SDN-based big data centers," in *Proc. IEEE 31st Int. Conf. Adv. Inf. Netw. Appl. (AINA)*, Mar. 2017, pp. 1106–1115.

[3] M. Kiran and A. Chhabra, "Understanding flows in high-speed scientific networks: A netflow data study," *Future Gener. Comput. Syst.*, vol. 94, pp. 72–79, May 2019.

[4] M. Noormohammadpour and C. S. Raghavendra, "Datacenter traffic control: Understanding techniques and tradeoffs," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 2, pp. 1492–1525, 2nd Quart., 2018.

[5] S. E. Gómez, B. C. Martínez, A. J. Sánchez-Esguevillas, and L. H. Callejo, "Ensemble network traffic classification: Algorithm comparison and novel ensemble scheme proposal," *Comput. Netw.*, vol. 127, pp. 68–80, Nov. 2017.

[6] X. Zhang, Y. Li, R. Kotagiri, L. Wu, Z. Tari, and M. Cheriet, "KRNN: K rare-class nearest neighbour classification," *Pattern Recognit.*, vol. 62, pp. 33–44, Feb. 2017.

[7] S. E. Gómez, L. Hernández-Callejo, B. C. Martínez, and A. J. Sánchez-Esguevillas, "Exploratory study on class imbalance and solutions for network traffic classification," *Neurocomputing*, vol. 343, pp. 100–119, May 2019.

[8] L. Ding, J. Liu, T. Qin, and H. Li, "Internet traffic classification based on expanding vector of flow," *Comput. Netw.*, vol. 129, pp. 178–192, Dec. 2017.

[9] Z. Liu, R. Wang, and M. Tao, "SmoteAdaNL: A learning method for network traffic classification," *J. Ambient Intell. Hum. Comput.*, vol. 7, no. 1, pp. 121–130, Feb. 2016.

[10] Z. Liu and Q. Liu, "Studying cost-sensitive learning for multi-class imbalance in Internet traffic classification," *J. China Universities Posts Telecommun.*, vol. 19, no. 6, pp. 63–72, Dec. 2012.
[11] H. Zhang, F. Tang, and L. Barolli, "Efficient flow detection and scheduling for SDN-based big data centers," *J. Ambient Intell. Hum. Comput.*, vol. 10, no. 5, pp. 1915–1926, May 2019.
[12] H. Rastegarfar, M. Glick, N. Viljoen, M. Yang, J. Wissinger, L. LaComb, and N. Peyghambarian, "TCP flow classification and bandwidth aggregation in optically interconnected data center networks," *J. Opt. Commun. Netw.*, vol. 8, no. 10, pp. 777–786, Oct. 2016.
[13] R. Hasibi, M. Shokri, and M. Dehghan, "Augmentation scheme for dealing with imbalanced network traffic classification using deep learning," 2019, *arXiv:1901.00204*. [Online]. Available: http://arxiv.org/abs/1901.00204
[14] A. Fernández, S. del Río, N. V. Chawla, and F. Herrera, "An insight into imbalanced big data classification: Outcomes and challenges," *Complex Intell. Syst.*, vol. 3, no. 2, pp. 105–120, Jun. 2017.
[15] Y. Sun, M. S. Kamel, A. K. C. Wong, and Y. Wang, "Cost-sensitive boosting for classification of imbalanced data," *Pattern Recognit.*, vol. 40, no. 12, pp. 3358–3378, Dec. 2007.
[16] S.-C. Chao, K. C.-J. Lin, and M.-S. Chen, "Flow classification for software-defined data centers using stream mining," *IEEE Trans. Services Comput.*, vol. 12, no. 1, pp. 105–116, Jan. 2019.
[17] S. del Río, V. López, J. M. Benítez, and F. Herrera, "On the use of MapReduce for imbalanced big data using random forest," *Inf. Sci.*, vol. 285, pp. 112–137, Nov. 2014.
[18] S. Ahmed, F. Rayhan, A. Mahbub, M. R. Jani, S. Shatabda, and D. M. Farid, "LIUboost: Locality informed under-boosting for imbalanced data classification," in *Emerging Technologies in Data Mining and Information Security*. Springer, 2019, pp. 133–144.
[19] C. Seiffert, T. M. Khoshgoftaar, J. Van Hulse, and A. Napolitano, "RUSBoost: A hybrid approach to alleviating class imbalance," *IEEE Trans. Syst., Man, Cybern. C, Appl. Rev.*, vol. 40, no. 1, pp. 185–197, Jan. 2010.
[20] A. Cano and B. Krawczyk, "Kappa updated ensemble for drifting data stream mining," *Mach. Learn.*, vol. 109, no. 1, pp. 175–218, Jan. 2020.
[21] J. Bian, D. Tian, Y. Tang, and D. Tao, "A survey on trajectory clustering analysis," 2018, *arXiv:1802.06971*. [Online]. Available: http://arxiv.org/abs/1802.06971
[22] J. Bian, D. Tian, Y. Tang, and D. Tao, "Trajectory data classification: A review," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 4, pp. 1–34, Aug. 2019.
[23] X. Shi, Y. D. Wong, M. Z.-F. Li, C. Palanisamy, and C. Chai, "A feature learning approach based on XGBoost for driving assessment and risk prediction," *Accident Anal. Prevention*, vol. 129, pp. 170–179, Aug. 2019.
[24] R. Soleymani, E. Granger, and G. Fumera, "Progressive boosting for class imbalance and its application to face re-identification," *Expert Syst. Appl.*, vol. 101, pp. 271–291, Jul. 2018.
[25] Z. Zhang, K. Huang, T. Tan, P. Yang, and J. Li, "ReD-SFA: Relation discovery based slow feature analysis for trajectory clustering," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 752–760.
[26] M. El Habib Daho, N. Settouti, M. E. A. Lazouni, and M. E. A. Chikh, "Weighted vote for trees aggregation in random forest," in *Proc. Int. Conf. Multimedia Comput. Syst. (ICMCS)*, Apr. 2014, pp. 438–443.
[27] F. Angiulli, "On the behavior of intrinsically high-dimensional spaces: Distances, direct and reverse nearest neighbors, and hubness," *J. Mach. Learn. Res.*, vol. 18, pp. 1–170, 2017.
[28] D. Yan, Z. Zhao, and W. Ng, "Monochromatic and bichromatic reverse nearest neighbor queries on land surfaces," in *Proc. 21st ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, 2012, pp. 942–951.
[29] Y. Tao, D. Papadias, X. Lian, and X. Xiao, "Multidimensional reverse kNN search," *VLDB J.*, vol. 16, no. 3, pp. 293–316, May 2007.
[30] S. Dawar, V. Goyal, and D. Bera, "Efficient reverse k nearest neighbor evaluation for hierarchical index," 2015, *arXiv:1506.04867*. [Online]. Available: http://arxiv.org/abs/1506.04867
[31] A. Barewar, M. A. Radke, and U. A. Deshpande, "Geo skip list data structure–storing spatial data and efficient search of geographical locations," in *Proc. Int. Conf. Adv. Comput., Commun. Informat. (ICACCI)*, Sep. 2014, pp. 1479–1485.
[32] D. Wu, X. Chen, C. Chen, J. Zhang, Y. Xiang, and W. Zhou, "On addressing the imbalance problem: A correlated knn approach for network traffic classification," in *Proc. Int. Conf. Netw. Syst. Secur.* Springer, 2015, pp. 138–151.
[33] S. S. Meriem Amina, B. Abdolkhalegh, N. K. Khoa, and C. Mohamed, "Featuring real-time imbalanced network traffic classification," in *Proc. IEEE Int. Conf. Internet Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phys. Social Comput. (CPSCom) IEEE Smart Data (SmartData)*, Jul. 2018, pp. 840–846.
[34] G. Casanova, E. Englmeier, M. E. Houle, P. Kröger, M. Nett, E. Schubert, and A. Zimek, "Dimensional testing for reverse k-nearest neighbor search," *Proc. VLDB Endowment*, vol. 10, no. 7, pp. 769–780, Mar. 2017.
[35] CAIDA. (2011). *Caida Aonymized 2011 Internet Traces Dataset*. [Online]. Available: http://www.caida.org/data/monitors/passive-equinix-chicago.xml
[36] T. Benson. (2010). *Data Set for IMC 2010 Data Center Measurement*. [Online]. Available: http://cs.brown.edu/~tab/
[37] UNIBS. (2009). *Unibs: Data Sharing*. [Online]. Available: http://netweb.ing.unibs.it/~ntw/tools/traces/index.php
[38] M. F. Iqbal, M. Zahid, D. Habib, and L. K. John, "Efficient prediction of network traffic for real-time applications," *J. Comput. Netw. Commun.*, vol. 2019, Feb. 2019, Art. no. 4067135.
[39] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th Annu. Conf. Internet Meas. (IMC)*, 2010, pp. 267–280.
[40] H. Zhang, G. Lu, M. T. Qassrawi, Y. Zhang, and X. Yu, "Feature selection for optimizing traffic classification," *Comput. Commun.*, vol. 35, no. 12, pp. 1457–1471, Jul. 2012.
[41] H. Doroud, G. Aceto, W. de Donato, E. A. Jarchlo, A. M. Lopez, C. D. Guerrero, and A. Pescape, "Speeding-up DPI traffic classification with chaining," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Dec. 2018, pp. 1–6.
[42] M. L. McHugh, "Interrater reliability: The kappa statistic," *Biochemia Medica*, vol. 22, no. 3, pp. 276–282, 2012.
[43] S. Jain, E. Kotsampasakou, and G. F. Ecker, "Comparing the performance of meta-classifiers—A case study on selected imbalanced data sets relevant for prediction of liver toxicity," *J. Comput.-Aided Mol. Des.*, vol. 32, no. 5, pp. 583–590, 2018.
[44] S. A. Shilbayeh, "Cost sensitive meta-learning," Ph.D. dissertation, Dept. Comput. Sci. Eng., Univ. Salford, Salford, WI, USA, 2015.
[45] J. Demšar, "Statistical comparisons of classifiers over multiple data sets," *J. Mach. Learn. Res.*, vol. 7, pp. 1–30, Jan. 2006.
[46] Z. Ali and S. Bhaskar, "Basic statistical tools in research and data analysis," *Indian J. Anaesthesia*, vol. 60, no. 9, p. 662, 2016.

MERIEM AMINA SI SABER received the B.Eng. degree (Hons.) in networking and telecommunications engineering from the University of Science and Technology Houary Boumedienne (USTHB), in 2013. She is currently pursuing the Ph.D. degree with the Synchromedia Laboratory for Multimedia Communication in Telepresence, University of Quebec, Montreal, under the supervision of Dr. M. Cheriet. She worked as a QoS and Capacity Planning Engineer with Telecom Algeria Mobilis, for almost three years. Her research interests include traffic engineering, QoS scheduling, and applied machine learning in data centers.

MEHDI GHORBANI received the bachelor's degree from the Sharif University of Technology. He is currently pursuing the master's degree in telecommunications networks with the Synchromedia Laboratory for Multimedia Communication in Telepresence, University of Quebec.

ABDOLKHALEGH BAYATI received the bachelor's degree in software engineering from the Shahid Chamran University of Ahwaz, in 2009, and the master's degree in information technology from the Sharif University of Technology, Tehran, in 2012. He is currently pursuing the Ph.D. degree with the Synchromedia Laboratory for Multimedia Communication in Telepresence, Department of System Engineering, University of Quebec, under the supervision of Prof. C. Mohamed. He worked on the project Error Concealment in Video. The goal of this project was to fix damaged motion vectors of video. He used statistical methods to model motion vector of video; and then, he estimated lost motion vectors. His main research interests are data science, machine learning, and pattern recognition. Most of his previous study and experiences can be summarized in statistics, data analysis, statistical data modeling, and signal processing. He has worked on different types of data such as image, video, and network traffic.

KIM-KHOA NGUYEN received the Ph.D. degree in electrical and computer engineering from Concordia University. He served as the CTO of Inocybe Technologies, a leading company in software-defined networking solutions. He was an Architect of the Canarie's GreenStar Network and also involved in establishing CSA/IEEE standards for green ICT. He has led research and development in large-scale projects with Ericsson, Ciena, Telus, and InterDigital. He is currently an Associate Professor with the Department of Electrical Engineering, École de technologie supérieure, University of Quebec, Montreal, QC, Canada. He has authored 60 publications and holds several industrial patents. His research interests include network optimization, cloud computing, the IoT, big data, machine learning, smart city, high-speed networks, and green ICT. He was a recipient of the Microsoft Azure Global IoT Contest Award, in 2017, and the Ciena's Aspirational Prize, in 2018.

MOHAMED CHERIET (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science from the University of Pierre et Marie Curie (Paris VI), Paris, France, in 1985 and 1988, respectively. Since 1992, he has been a Professor with the Department of Automation Engineering, École de Technologie Supérieure (ÉTS), University of Quebec, Montreal, QC, Canada, where he was a Full Professor, in 1998. Since 1998, he has been the Founder and the Director of the Synchromedia Laboratory for Multimedia Communication in Telepresence, (ÉTS), which targets multimedia communication in telepresence applications. He also Co-Founded the Laboratory for Imagery, Vision, and Artificial Intelligence, ÉTS, where he was the Director, from 2000 to 2006. He is an expert in computational intelligence, pattern recognition, mathematical modeling for image processing, cognitive learning, and machine learning approaches and perception. His research has acquired extensive experience in cloud computing and network virtualization. He has authored or coauthored more than 450 technical articles in the field. He has coauthored a book *Character Recognition Systems: A Guide for Students and Practitioners* (Wiley, Spring 2007). He was a recipient of the 2016 IEEE J. M. Ham Outstanding Engineering Educator Award and the 2012 Queen Elizabeth II Diamond Jubilee Medal. He is a fellow of the International Association for Pattern Recognition, the Canadian Academy of Engineering, and the Engineering Institute of Canada, the Founder and the Former Chair of the IEEE Montreal Chapter of Computational Intelligent Systems, the Steering Committee Member of the IEEE Green ICT Initiative, and the Chair of the IEEE ICT Emissions Working Group. He serves on the editorial boards of several renowned journals and international conferences.