Information Sciences 687 (2025) 121407

Contents lists available at ScienceDirect

Information Sciences

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

Optimisation of multiple clustering based undersampling using artificial bee colony: Application to improved detection of obfuscated patterns without adversarial training

Tonkla Maneerat [a], Natthakan Iam-On [b], Tossapon Boongoen [b,*], Khwunta Kirimasthong [a], Nitin Naik [c], Longzhi Yang [d], Qiang Shen [b]

[a] Center of Excellence in AI and Emerging Technologies, School of Information Technology, Mae Fah Luang University, Chiang Rai, Thailand
[b] Advanced Reasoning Research Group, Department of Computer Science, Aberystwyth University, Aberystwyth, UK
[c] School of Informatics and Digital Engineering, Aston University, Birmingham, UK
[d] Department of Computer and Information Sciences, Northumbria University, Newcastle, UK

ARTICLE INFO

Keywords:
Adversarial attack
Intrusion detection
Classification
Class imbalance
Ensemble clustering

ABSTRACT

Attack detection is one of the main features required in modern defence systems. Despite the ongoing research, it remains challenging for a typical mechanism like network-based intrusion detection system (NIDS) to catch up with evolving adversarial attacks. They specifically aim to confuse a machine-learning based predictor. Without the knowledge of adversarial patterns, the best approach is generalising signatures learned from a dataset of legitimate connections and known intrusions. This work focuses on analysing non-payload traffics so that the resulting techniques can be exploited to a range of network-based applications. It investigates a novel means to deal with the problem of imbalanced classes. An optimised undersampling method is introduced to select a subset of majority-class representatives initially created through an ensemble clustering procedure. A weighted combination of criteria representing distributions within and between classes is proposed as the objective function for a global optimisation using the artificial bee colony (ABC). This approach usually outperforms its baselines and other state-of-the-art undersampling models, with ABC being more effective using the global best strategy than a random selection of solutions or an iterative greedy search. The paper also details the parameter analysis offering a heuristic guide for potential taking up of the proposed techniques.

1. Introduction

In the era of digital world with an exponential increase of computing applications and network coverage, individuals and organisations are at risk of one or several forms of cyber-attack. This is brought about by online interactivity and personalised services such as mobile banking, intelligent control platform, a wide exploitation of Internet-of-Things or IoT and wireless sensor systems. It is essential, and urgent, to address the issue of network security and defence mechanism, especially when cybersecurity threat is highly adaptive in nature [1]. As a response, many approaches to facilitate the detection of intrusive traffics have been introduced to promptly trigger countermeasures. Among these, an intrusion detection system or IDS is a significant tool to ensure availability and

* Corresponding author.
E-mail address: tob45@aber.ac.uk (T. Boongoen).

https://doi.org/10.1016/j.ins.2024.121407
Received 15 February 2024; Received in revised form 12 June 2024; Accepted 25 August 2024

Available online 29 August 2024

integrity of networks [2]. It has proven effective for handling threats caused by uses of unpatched services, to which attackers typically deploy malicious codes. Importantly, a network-based IDS (NIDS) successfully safeguards online monitoring of complex systems, especially those related to critical national infrastructure. In particular, NIDS improves the host-based counterpart by expanding the scope of traffic analysis to the network level, whilst giving useful information for an efficient intervention by human operators.

Signatures of intrusions evolve rapidly, often in the effort to avoid a positive match. A new generation of NIDS therefore needs to be resilient to mutated patterns through polymorphism or obfuscation [3]. Obviously, a simple signature-based system would fail to catch up with those almost unseen connections, e.g., ones employed in a zero-day incident. A better alternative is machine learning (ML)-based NIDS, which can handle the aforementioned difficulty to a certain extent, albeit depending on the recency and quality of the training data available. In fact, several classification methods have been studied for this task, normally referred to as anomaly-based detection [4], including both supervised and unsupervised approaches of artificial neural networks (ANNs), support vector machine (SVM), ensemble of classifiers, and k-means. In spite of the ability to recall unknown patterns that partly resemble known attacks, these techniques fall short for recognising those obfuscated connections exploited in an adversarial attack. This creative breed of intrusion aims to confuse ML-based NIDS, where perturbations of traffic data are adopted as new connections. As such, a pre-trained model may wrongly determine a decision boundary and prediction.

As it may be difficult to acquire or even to guess exact obfuscation techniques used by attackers, sensible solutions are to i) generate a potential collection of adversarial patterns to train an ML-based detection model (i.e., adversarial training [5]), ii) keep training the classifier with new training data, including mutated instances by newly updated obfuscation techniques, but with a chance of being the victim of data poison attacks [6], and iii) refine the stages in developing a target model (e.g., data pre-processing or classification modelling) from conventional data sets such that the resulting model is generalised to new and modified connection samples [7]. Without the knowledge of adversarial samples, the current research follows the third category with an assumption that a detection model can be strengthened by generalising the learning procedure, focusing more on the intrusion class that would usually be a minority in most datasets. Considering this, the problem of class imbalance has been tackled in the literature, where synthetic minority oversampling or SMOTE and its modifications are explored to leverage sample sizes. Despite simplicity and good training performance, they often fail to generalise to new data due to the curse of overfitting. The task may be better handled by an undersampling strategy but with a possible sacrifice of some informative samples [8]. Inspired by this insight and initial investigations that show the potential of clustering methods for undersampling [7], the current work extends this line of research by improving the mechanism for the selection of representative instances, defined as a nature-inspired optimisation process. As a result, the chosen data preserves main characters of the underlying majority class, whilst simplifying the borderline between classes.

Assumption: Based on recent works [3,7], this research concentrates on classification modelling for NIDS, operating solely with non-payload traffic data that is represented as a set of connections at the TCP layer. It is assumed that adversarial attackers are only able to make changes to system inputs using obfuscation methods, provided that they remain conforming to the standard TCP/IP protocol. In particular, known intrusive signatures are exploit-independently modified to resemble normal connections. Within the taxonomy of adversarial attacks to IDS, this work belongs to evasions that target the measurement phase.

Problem and Scope: Following [7] that proposes a multiple-clustering method to analysing a benchmark data collection, this study extends the framework by defining the selection of representative instances as a global optimisation problem, which is resolved using swarm intelligence. This new extension may achieve a better set of representatives than the baseline models that commonly exploit a greedy search. Note that these are chosen from a pool of multiple clusterings produced along the concept of cluster ensemble [9]. For the binary classification problem with majority and minority classes corresponding to legitimate and attack connections, the size of majority is reduced to be in par with that of the other. Hence, the resultant target classifier can learn more from the minority class, i.e., a better chance to identify mutated patterns. At the same time, clustering-wise data may also help to reduce information loss in the majority. Consequently, without the knowledge of adversarial attack patterns, the detection rate of these malicious signatures may be improved by a classification model trained only on a standard dataset.

Contributions: Firstly, this work extends the initial studies of [3,7], creating a new undersampling method to support the development of classification-based NIDS on standard legitimate-known attack datasets, which can be more effective to detect adversarial intrusions. Building on the multiple-clustering framework, it presents a new selection strategy as a nature-inspired global optimisation, instead of a simple greedy search introduced in the existing work. This unique combination is generalised to prepare balanced training data for subsequent adaptation by any feature-based ML algorithms.

Secondly, given a pool of cluster centroids generated from clustering instances of the majority class using the consensus clustering [9], a new optimisation procedure is introduced to select a subset of these representatives, whose property optimises the target fitness function. This is designed as a single objective criterion that integrates two distinct aspects of a good undersampling set: (i) the diversity among centroids in the chosen subset, and (ii) the diversity between this subset and samples belonging to the minority class. The integration is weighted to provide flexibility for facilitating further analysis and applications to different domain problems. In particular, the ABC optimisation algorithm [10] is employed to perform a global/meta-heuristics search, with two new operations that allow potential solutions to evolve. These are included in three different phases conducted by employee, onlooker and scout bees. Also, a random selection is used to identify a target food source for onlooker bees during the exploitation stage. A comparison between this and the global best strategy [11] is also provided. Note that ABC has been exploited in the field of NIDS for classification modelling [12], feature selection [13] and data sampling [14] among others. Despite these, the proposed application of ABC in conjunction with ensemble clustering and undersampling is new and can be useful for other classification tasks.

Lastly, performance of the newly developed model is assessed and compared to a wide range of possible alternatives, including: its baselines (initial multiple-clustering [7] and single-clustering [15] models), Random UnderSampling (RUS) [16], RUSBoost [16] and Inverse Random UnderSampling (IRUS) [17]. These also include recent undersampling methods: Membership Probability-based

Undersampling (MPU) [18], two-stage undersampling (DBIG-US) [19] and Cluster-based Instance Selection (CBIS) [20]. To complete the picture, Outlier-SMOTE [21] as a new variation of SMOTE is also examined. Furthermore, as a guideline for users, parameter analysis is provided to emphasise the relation between model performance and its underlying variables.

Organisation of this paper is summarised as follows. In Section 2, problem definition, background and the dataset examined in this research are presented. Then, the proposed multiple-clustering undersampling model and the method for its optimised selection of representatives are provided in Section 3. Section 4 discusses the performance evaluation with respect to different experimental settings. Lastly, 5 concludes the paper with a highlight of potential future investigation.

2. Related works

To provide background for this study, related works on handling the imbalance issue in a classification-based NIDS are briefly reviewed. This section also includes information on the specification of traffic objects, obfuscation concept and relevant techniques.

2.1. ML-based NIDS and handling of imbalanced classes

The demand of an accurate NIDS has been high given the advancement and popularity of wireless sensor networks and IoT applications. It can be perceived as an intelligent agent that constantly monitors outgoing and incoming traffics for connections that may be a part of an intrusion. At large, NIDSs acquire this capacity through a simple matching of new connections to known legal signatures. However, such an approach suffers from a low recall as the list of legitimates may miss new entries from recent incidents. Many attempts have been made to overcome this using a predictive model that is built from up-to-date data. An alternative solution is to exploit the concept of anomaly detection in order to identify traffics that are significantly dissimilar to others [4]. Nonetheless, it has the drawback of a high false-positive rate because the determination of borderline between legitimate and intrusive acts is not straightforward.

Another approach to this quest is witnessed with many ML-based NIDSs found in the literature. In particular, many classifiers have demonstrated desirable behaviours as they explore strength of both signature-matching and anomaly-led ideas. To begin with, different classification algorithms are exploited to develop a predictive model that accommodates minority attack classes [4]. The SMOTE technique is specifically employed to increase the cardinality of evasive instances belonging to those groups. A similar study has recently been reported by [22], in which deep learning models are included in addition to those conventional ML methods. Further details related to these ideas are summarised in Table 1, while a boarder umbrella of NIDS research can be found in the reviews of [23].

Particularly, the first nine techniques listed in Table 1 are generally capable of tackling the presence of imbalance class distribution in training data. Basic data sampling methods like SMOTE, RUS or their extensions are exploited to prepare a desired dataset to develop classification models, ranging from benchmark feature-based approaches to deep learning methods [7,24–27]. In addition to this first category, the ABC algorithm has also been exploited to obtain a balanced dataset, see the study of [14] for an example. Unlike those emphasised above, many other methods specified in Table 1 resolve the imbalance problem implicitly by manipulating feature engineering techniques, e.g., by seeking an optimal attribute subset [3,28–32] or a mixture of data sampling and feature selection [33,34]. Again, ABC is also adopted for this feature selection task for NIDS. For instance, a hybrid model introduced by [35] makes use of ABC and the artificial fish swarm technique to optimise both fuzzy c-means and correlation-driven feature selection processes within the data preparation stage. Similarly, ABC has been employed to extract important features prior training an AdaBoost classifier [13]. Besides, ABC provides a way to optimise algorithmic parameters for different classification models developed in the NIDS field, e.g., ANN and LR [12].

The concept of ensemble has also been exploited to generate a system of multiple classifiers, each of which learns from one of many data perturbations [13]. This is further elaborated by [36,37], where a feature selection function is integrated within the ensemble learning framework. Another interesting progress is made to obtain a high precision using simple oversampling and undersampling methods to produce classifiers for the voting-based ensemble [22]. These developments have revealed the significance of class imbalance to improving predictive performance of NIDS, especially for the challenge of obfuscation [3]. Together, these approaches motivate the current work with the focus on handling imbalanced classes in NIDS.

2.2. Adversarial attack to ML-based NIDS

Apart from the subject of NIDS, adversarial attacks have also been investigated in domains such as natural language processing, image processing and computer vision, and in detection of malware. Unlike the techniques that modify features already extracted from raw data, the focus of attacks against ML-based NIDS is to exploit a mutation of original traffic data, which is difficult for a simple detector to realise straight away. Following this observation and the recognition that an attacker typically possesses knowledge of the target NIDS, a so-called white-box attack has been proposed with several mutation functions [38]. In practice, such an approach is not easily attainable, however. Other studies including the work presented in this paper exclude the prior knowledge assumed. Instead, they disguise attack connections using traffic morphing and obfuscation techniques [39].

In accordance with the taxonomy adopted in the literature [40], two main approaches exist to modify traffic data by considering either non-payload or payload-based part of connections. For the former, a mutation of HTTP request is deployed to weaken NIDS, while malware morphism techniques are explored to alter the traffic payload. These methods have been criticised for being inefficient and not applicable to different network settings, thus leading to a series of works in the non-payload counterpart. Recent studies

Table 1
Related works for ML-based NIDS and handling of class imbalance. Abbreviations are PCA: principal component analysis, DNN: deep neural networks, RF: random forest, LDA: linear discriminant analysis, KNN: k-nearest neighbours, DT: decision tree, KM: k-means clustering, SVM: support vector machine, ANN: artificial neural network, and LR: logistic regression.

Existing works	Techniques exploited	Handling of imbalanced classes
Online PCA based oversampling [25]	PCA	oversampling
Balancing data for NIDS [24]	DNN & RF	under- & oversampling
ML modelling for imbalance data [4]	LDA, RF, KNN & DT	oversampling
Hybrid ML framework [33]	KNN, KM & feature selection	oversampling
Information-gain feature modelling [34]	RF	oversampling
Auto-encoder oversampling [26]	RF, SVM, ANN & LR	oversampling
Adversarial reinforcement learning [27]	DT, SVM, ANN, NB, KNN & LR	oversampling
Voting based ensemble modelling [22]	DT, NB, SVM, RF, LR & ANN	over- & undersampling
Multiple-clustering undersampling [7]	SVM, NB, DT & LR	undersampling
Adaptive ensemble modelling [36]	DNN, KNN, RF & DT	n/a
Filter-based feature elimination [29]	SVM & KM	n/a
Recursive feature reduction [31]	RF, DT & SVM	n/a
Feature reduction method [28]	ANN	n/a
RF modelling for NIDS [30]	RF & feature selection	n/a
Forwarding feature selection [3]	SVM, NB, DT & LR	n/a
ANN-Bayesian based feature selection [32]	ANN & Bayesian net	n/a
Classifier ensemble for IDS [37]	DT, RF & feature selection	n/a
Classifier ensemble for NIDS [13]	AdaBoost & DT	n/a

have also provided new findings from an investigation into the performance of various classification techniques on a simulated data collection [3], treating up-to-date threats and obfuscation methods conjunctively. Particularly, model accuracies are enhanced through a wrapper search for informative subset of features, and the resulting framework is extended to cover the problem of imbalanced classes [7]. Furthermore, a multiple-clustering based undersampling has been introduced to reduce the size of majority class, where a greedy procedure is in place for the selection of representative samples from a pool of cluster centroids. Of course, such a search mechanism is often suboptimal, thus opening the research gap examined in this paper.

Traffic connections and non-payload obfuscation are key concepts in the present work, where data objects of interest correspond to TCP connection instances that are interchanged between client and server parties, summarising communication packets up to the transport layer of common TCP/IP stack. Various attributes can be used to represent a connection λ, including timestamp, client/server IP address and port number, or any other identifiable property embedded in the packets sent back and forth between both ends.

Formally, a function $f_i(\lambda) \in F(\lambda), i = 1 \dots d$ maps a connection λ to the i-th feature γ_i within a feature space $\Gamma = (\gamma_1, \gamma_2, \dots, \gamma_d)$. Based on the investigation reported in the literature, examples of such features are length of packet header, inbound and outbound packet sizes, numbers of TCP finish and TCP urgent flags raised for inbound traffics. Let a connection λ^a denote a direct attack involving d features of $\gamma_1^a, \gamma_2^a, \dots, \gamma_d^a$. The idea of obfuscation is to modify λ^a so that a variation $\lambda^{a'}$ is obtained, whose non-payload attributes $\gamma_1^{a'}, \gamma_2^{a'}, \dots, \gamma_d^{a'}$ are the results of removing, inserting, or transforming the original content of underlying packets respectively. In simulating a realistic threat dataset exhibiting actual attack types, different mutation functions have been introduced. These include partial losses of packet data, modifications of packet order, delays of packet sending, different fragmentation models, errors caused by an unreliable network, or any combination of these [3], setting up the foundation for the following development.

3. Proposed approach

This section describes the proposed framework that extends the recent undersampling model based on multiple clusterings [7]. The problem is first formalised, which is followed by a detailed description of the proposed solution mechanism.

3.1. Problem statement

Following previous works, the predictive capability of NIDS developed here is acquired by classification modelling. Let $T = X \times Y$ represent a training data set, where X denotes the feature space Γ of n connections, $X \in R^{n \times d}$, and Y of the size $n \times 1$ corresponds to a set of class labels, with a connection $x_i \in X$ being categorised as $y_i \in Y$. Note that the value assigned to y_i is taken from the domain D_T of target classes. Generally, given a preferred classification algorithm ω and a data set T, a trained classifier $CSF_{T\omega}$ is utilised to determine a class solution $y_{new} \in D_T$ of a new connection $x_{new} \in R^{1 \times d}$, i.e., $CSF_{T\omega}(x_{new}) = y_{new}$. As such, the question dealt with within this work is how to perform the binary classification of a connection λ as $y_\lambda \in D_T = \{attack, legitimate\}$. Similarly, the following equation specifies this relation specific to the case of a direct attack λ^a.

$$y_{\lambda^a} = CSF_{T\omega}(F(\lambda^a)) \tag{1}$$

With the application of non-payload obfuscation techniques, the focus has shifted to maintaining the accuracy of $y_{\lambda^{a'}}$, which is generated by $CSF_{T\omega}$ over the feature space $F(\lambda^{a'})$ of a mutated connection $\lambda^{a'}$. As a response to the challenge of recognising novel obfuscated instances, a feature selection method is exploited to boost the predictive performance of classification model that is trained

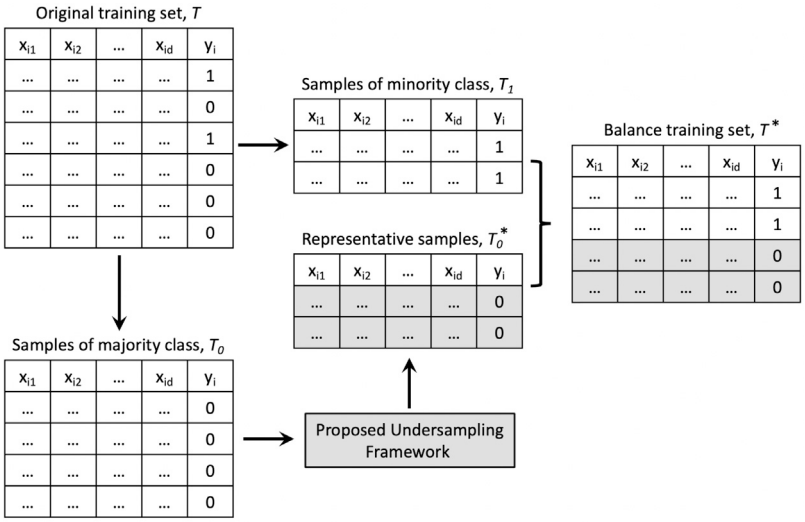

Fig. 1. Proposed framework that reduces training set T to T^*.

only with legitimate connections and direct attacks. This follows the initial studies reported in [3,7] and is in line with existing works identified in Table 1. The problem of class imbalance is treated as a burden to achieve a resilient predictor, and a multiple-clustering undersampling approach is introduced prior to the training phase. Importantly, rather than selecting representative instances for the majority class by simply using a greedy search [15], which may lead to a suboptimal result, a new optimisation framework is proposed in the following.

3.2. Solution method

Instead of selecting representative samples of the majority class from a single clustering, they can be chosen from a pool of diverse candidates created by the use of consensus clustering. Different from existing works, the selection process is herein devised as a global optimisation problem. Having taken notice of the efficacy of artificial bee colony (aka., ABC) [10] in performing global optimisation tasks, ABC is utilised to maximise the pre-defined fitness function. Key stages of the proposed approach are detailed below, including the initial generation of a candidate pool, the optimised selection of representative instances, and the subsequent development of an effective classification system.

3.2.1. Generating pool of representative candidates

Suppose that a training data set $T = X \times Y$ consist of a set T_1 of samples belonging to the minority class and another T_0 of those to the majority class. Let $T_1 = X_1 \times Y_1$ and $T_0 = X_0 \times Y_0$ be feature and label spaces of the two groups, respectively. Unlike an oversampling approach that aims to increase the sizes of T_1, an undersampling method β is set to extract a subset of representative samples from T_0, i.e., $\beta(T_0) = T_0^*$ where $|T_0^*| \ll |T_0|$ with $|A|$ being the size of set A. From that, the desired training data T^* without the presence of class imbalance is constructed as $T^* = T_1 \cup T_0^*$. Based on the choice of algorithm ω, it is then employed to develop a classifier $CSF_{T^*\omega}$, which is typically evaluated via comparison to the model $CSF_{T\omega}$ trained from the original data set.

Initially, the extraction function β is designed to determine the centroids Z from a single clustering on X_0, without prior knowledge of the labels in Y_0. At this stage, the specific number of clusters $\rho = |T_1|$, and the resulting centroids $z_1, z_2, \ldots, z_\rho$ are taken as the representative samples. As recognised in the literature [7], the quality of this pool can be improved by multiple clusterings, from which the same number of ρ centroids are selected. Specific to the current work, let $X_0 = \{x_1, \ldots, x_{n_0}\}$ denote the feature space in the normalised domain $[0,1]^{n_0 \times d}$ with n_0 and d representing the number of legitimate connections and that of features. Let each $x_i \in X_0$ be a vector of d feature values or $x_i = (x_{i1}, \ldots, x_{id}), \forall i \in \{1, 2, \ldots, n_0\}$, while $\Pi = \{\pi_1, \pi_2, \ldots, \pi_M\}$ represents an ensemble of M clusterings. For the α-th ensemble member, a data partition containing k_α is produced, i.e., $\pi_\alpha = \{C_1^\alpha, \ldots, C_{k_\alpha}^\alpha\}$, provided that $\bigcup_{s=1}^{k_\alpha} C_s^\alpha = X_0$ and $\bigcap_{s=1}^{k_\alpha} C_s^\alpha = \emptyset$. This procedure is illustrated in Fig. 1.

To ensure diversity among members of Π, the following generation schemes are adopted:

- Random-k strategy: For each clustering, the number of clusters is arbitrarily taken from the range of $\{2, \ldots \sqrt{|X_0|}\}$ or $\{2, \ldots L\}$ when $L < \sqrt{|X_0|} < (L+1)$, L being an integer greater than 2. This is used in addition to the random initialisation of k-means. Note that L is practically set to 50 (to signify a significantly large number of clusters being considered) unless otherwise stated.
- Random-subspace strategy: A clustering $\pi_\alpha, \alpha = 1 \ldots M$ is produced from a subset of X_0, or a feature subspace hereafter. This is perceived as a random subset $X_0' \in [0,1]^{n_0 \times d'}$ of the original feature space X_0, whose number of chosen features is estimated by

$$d' = d'_{min} + \lfloor \eta(d'_{max} - d'_{min}) \rfloor, \tag{2}$$

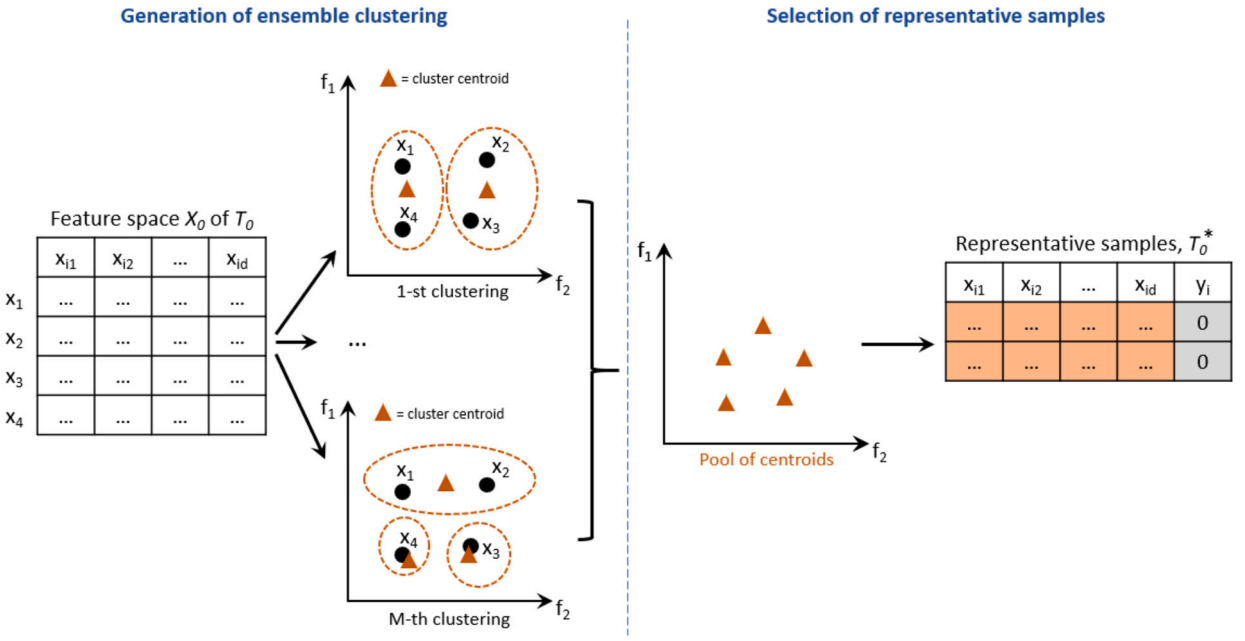

Fig. 2. Two stages of multiple-clustering based undersampling: generation of ensemble clustering and selection of representative centroids to construct T_0^*.

where $\eta \in [0,1]$ is a uniform random number, d'_{max} and d'_{min} stand for upper and lower limits of a subspace X'_0. As with the practice in the literature, d'_{max} and d'_{min} are set to $0.85d$ and $0.75d$, without duplicated features appearing in X'_0.

3.2.2. *Selecting representative samples*

Using the above strategies for the generation of initial representative candidates, a cluster ensemble Π is produced with the corresponding pool Z_Π of cluster centroids. Each of the resulting $z_q \in Z_\Pi$ is summarised from a cluster $C_q \in \{C_1^1, C_2^1, \ldots, C_{k_1}^1\} \cup \{C_1^2, C_2^2, \ldots, C_{k_2}^2\} \cup \ldots \{C_1^M, C_2^M, \ldots, C_{k_M}^M\}$, and it is encoded as a vector of original d dimensions $z_q \in [0,1]^{1\times d}$. The purpose of this stage is to select a collection $Z = \{z_1, z_2, \ldots, z_\rho\}$ of ρ centroids from all ρ' candidates in the current pool to express the feature space X_0^* of representative samples. Note that the label space Y_0^* is initially provided with all entries having the same value of 0, while $\rho' = k_1 + k_2 + \ldots + k_M$. See Fig. 2 for the illustration of this processing stage.

This identification of Z is designed as an optimisation problem, which takes into account two viewpoints of quality exhibited by members of Z. Firstly, the idea of furthest-first is adopted to derive the following fitness function Ω_1 that prefers members of Z to be largely different from other candidates in the pool Z_Π.

$$\Omega_1 = \underset{\forall Z \subset Z_\Pi}{\arg\max} \frac{\sum_{\forall z_e \in Z} \sum_{\forall z_a \in Z_\Pi, z_a \neq z_e} d(z_e, z_a)}{|Z|(|Z_\Pi| - 1)} \tag{3}$$

For the second fitness function Ω_2, the same furthest-first approach is similarly applied to identify the set Z whose members are dissimilar to instances of the minority class.

$$\Omega_2 = \underset{\forall Z \subset Z_\Pi}{\arg\max} \frac{\sum_{\forall z_e \in Z} \sum_{\forall x_j \in X_1} d(z_e, x_j)}{|Z||X_1|} \tag{4}$$

Intuitively, both criteria are required to ensure the degree of diversity between those representative samples and to reduce a possible class overlapping at the borderline. Therefore, the following function is proposed to integrate these preferred characteristics including a flexibility of weighting to acquire a range of optimised results, such that

$$\Omega^* = \underset{\forall Z \subset Z_\Pi}{\arg\max}\ \kappa\Omega_1 + (1-\kappa)\Omega_2, \tag{5}$$

where $\kappa \in [0,1]$ is a weighting factor assigned to those two basic assessment criteria. Obviously, Ω^* may behave as Ω_1 if $\kappa = 1$, or Ω_2 as κ becomes zero. Thus, it helps accommodate this optimisation process to suit different assumptions regarding the importance degrees of intra-class sample distribution and inter-class overlapping. In particular, for the case that they are regarded equally important, $\kappa = 0.5$.

Having established this single fitness function, a greedy additive method may be used to repeatedly select the best candidate to join a target set Z. Despite its simplicity, a sub-optimal (or locally best) result may well be acquired at times using such a strategy

as indicated previously. This work aims to exploit swarm intelligence to enable global optimisation. In particular, ABC that has been successfully applied to many problem domains is adopted here. The design of ABC based optimisation of finding the best subset Z of Z_{Π} that maximises Ω^* can be explained as follows.

Step 1: Initialise algorithmic parameters specific to ABC, i.e., fn for the number of food sources in the population, each of which represents a possible solution of the underlying optimisation problem; $maxCyc$ for the maximum number of iterations that food sources evolve; and lmt for the maximum number of generations before a food source is dropped as it continues to show no improvement. Note that the last variable is exploited to diversify the underlying search.

Step 2: Randomly generate an initial set FS of fn food sources, each of which is encoded as a vector $fs_l \in \{0,1\}^{1\times\rho'}, l = 1 \ldots fn$. A value '1' at the j-th entry of fs_l or the j-th centroid in Z_{Π} is selected as a member of the target set Z, '0' otherwise. As a result, there will be only ρ, i.e. the size of Z, entries in fs_l with the value of 1. Note that there are no duplicated vectors in the resulting FS, and that these food sources are also recorded in a $tabuList$ that is used to prevent a generation of existing food sources later on. The fitness values of these food sources are estimated by the use of Ω^* and saved for subsequent stages. The life time $life_l$ of each food source fs_l is initially set as 0 to specify that it is new and yet to evolve. During the exploitation stages of ABC conducted by employee and onlooker bees, $life_l$ may be repeatedly incremented to the maximum of lmt before it is dropped from a population FS. This is the case when there is no alternative food source in the vicinity of the one examined by those two types of bees exhibits a better fitness value. Then, as part of the exploration stage of ABC, it is replaced by a new food source identified by a scout bee.

Step 3: Having initialised the population and parameters, the sequence of operations explained in Steps 3 to 6 is repeated to allow food sources to evolve, thus improving the quality of solutions. As mentioned above, three types of bees involve in this process: employee and onlooker bees for the exploitation of existing food source to find better alternatives, and scout bees for the exploration of a new food source, respectively. Specific to this step, send fn employee bees to investigate the current set of food sources FS, with the number of employee bees equaling to that of food sources. Each bee attempts to find a better alternative to a particular food source to which it is assigned. For $fs_l \in FS$, a variant fs'_l of fs_l is created using the neighbour-based member replacement algorithm, which is summarised in Algorithm 1. It cannot be a duplicate of any food source in $tabuList$, with itself being added to the list afterward. The existing food source will be replaced by a new one if its fitness value is smaller than that of the new, and the life time is reset $life_l = 0$; else, the food source remains the same with its life time $life_l$ being increased by one.

Step 4: After a possible update of FS, those employee bees return to the hive with information of food-source-specific fitness values. Next, one potential food source $fs^* \in FS$ is chosen to be further exploited by onlooker bees. In particular, fn of these bees are to further create alternatives of fs^* in the quest of enhancing its fitness through an evolution. Based on recent studies such as [11], two different selection operators are employed in this work: global-best and proportional. It is straightforward for the use of the first operator as it selects the food source with the highest fitness value, and randomly picks one if a tie happens. For the proportional method, fs^* is determined by a random process. Basically, this selection works by firstly translating all fn fitness values to areas of a circular wheel, i.e., a higher value occupies a bigger share of 360 degrees, or a higher chance to be chosen. Then, one of these (also the food source it represents) is picked up based on a random selection of angle between 0 and 360.

Step 5: Send fn onlooker bees to exploit the chosen food source fs^*, with each of these bees generating a variation of fs^* via applying the same algorithm used by the employee bees. Once such newly resultant food sources are checked for a possible duplication against each other and those in $tabuList$, they are added to this list and the one fs' with the highest fitness value is compared to fs^*. If the fitness value of the new food source fs' is greater than that of the original fs^* its life time is set to zero. Otherwise, there is no change made to this food source except its life time is incremented, indicating that there is no improvement in this generation.

Step 6: Send one or more scout bees to explore new alternatives. For those food sources with life time values exceeding the upper bound of lmt, they will be visited again by scout bees that replace those existing with new food sources. This is accomplished using the operator of inverse member selection, which reaches out for a largely different alternative, as detailed in Algorithm 2. Of course, a new food source is checked against $tabuList$ and appended to this list with its life time value reset to zero.

Step 7: Repeat Steps 3 to 6 inclusive, for $maxCyc$ iterations. Then, return the food source with the highest fitness value that corresponds to the target set Z of representative samples.

3.2.3. Developing classification system

The optimal set of centroids Z obtained from the preceding stage is exploited as the feature space $X_0^* \in [0,1]^{\rho\times d}$ of the majority class $Y_0^* = \{1\}^{\rho\times 1}$, i.e., $T_0^* = X_0^* \times Y_0^*$. Then, the desired training set is a combination $T^* = T_1 \cup T_0^*$, on which a classifier $CSF_{T^*\omega}$ can be trained, using the choice of algorithm ω. If desired, alternative methods available in the literature may also be used to achieve a balance training set, including those that are utilised in the following comparative empirical evaluations.

4. Performance evaluation

This section presents the results of performance evaluation of the proposed and compared systems, including data investigated, experimental setup, and a discussion of findings.

4.1. Experimental data investigated

The dataset examined in this work is taken from the literature [3] that aims to develop a benchmark resource for applying ML-based techniques to detecting non-payload adversarial attacks, covering feature selection-based approach and clustering-based

Algorithm 1 Neighbour-based member replacement.

FS, collection of fn food sources;
$fs_l \in FS$, current food source explored by employee bee;
fs'_l, new variation of current food source;
tabuList, list of all food sources explored so far;
neighbourList, list of neighbours already explored;
found, boolean variable to control repetitions.

(1) $neighbourList \leftarrow \emptyset, found \leftarrow False$
(2) **Repeat if** $found = False$
(3) Randomly select neighbour $fs_n \in FS, fs_n \neq fs_l$, $fs_n \notin neighbourList$
(4) $neighbourList \leftarrow fs_n$
(5) **For** $j = 1 \dots \rho'$
(6) $fs'_l = fs_l$
(7) **If** $fs'_l[j] = 1$
(8) $fs'_l[j] \leftarrow 0$
(9) **For** $g = 1 \dots \rho'$
(10) **If** $fs_n[g] = 1, fs'_l[g] = 0, g \neq j$
(11) $fs'_l[g] = 1$
(12) **If** $fs'_l \notin tabuList$
(13) $tabuList \leftarrow fs'_l$
(14) $found \leftarrow True$
(15) **break**
(16) **If** $found = True$
(17) **return** fs'_l
(18) **Else return** fs_l

Algorithm 2 Inverse member selection.

FS, collection of fn food sources;
$fs_l \in FS$, current food source explored by employee bee;
fs'_l, new variation of current food source;
tabuList, list of all food sources explored so far;
count, number to selected members;
found, boolean variable to control repetitions.

(1) $found \leftarrow False$
(2) **Repeat if** $found = False$
(3) $fs'_l \leftarrow \{0\}^{1\times\rho'}$
(4) $count \leftarrow 0$
(5) **Repeat if** $count < \rho$
(6) Randomly select $j \in \{1, \dots, \rho'\}$
(7) **If** $fs_l[j] = 0, fs'_l[j] = 0$
(8) $fs'_l[j] = 1$
(9) $count \leftarrow count + 1$
(10) **If** $fs'_l \notin tabuList$
(11) $tabuList \leftarrow fs'_l$
(12) $found \leftarrow True$
(13) **If** $found = True$
(14) **return** fs'_l
(15) **Else return** fs_l

undersampling method [7]. Initially, traffic instances corresponding to normal connections and attacks are collected from different network services, including Apache Tomcat, PostgreSQL, MsSQL, and Samba.

Fig. 3 depicts the service-specific distribution of the data acquisition process, with the categorisation of 11,445 samples into three classes of legitimate, direct and obfuscated attacks, respectively. To generate those obfuscated-attack connections, various obfuscation techniques are applied to mutate some of those direct attacks. After that, feature extraction is applied to the entire dataset, using Advanced Security Network Metrics (ASNM) [3] to transform traffic- to feature-level descriptors. Through a min-max normalisation, the resulting data consists of a normalised feature space $X \in [0, 1]^{11,445\times194}$, where a class $y_i \in Y$ of $x_i \in X$ is drawn from the domain of 3 labels, i.e., $y_i \in$ {Legitimate, Direct attack, Obfuscated attack}.

As with the practice in the literature [3], only 14 out of the 194 extracted features that highly correlate to classes of legitimate and direct attacks are exploited in the current evaluation. Note that this feature subset is found without the knowledge of those obfuscated connections. As a result, the feature space of this dataset becomes $X \in [0, 1]^{11,445\times14}$. Particularly, among the five types of the ASNM features, four statistical ones are selected, with the other 10 belonging to the behavioural category. Examples of the statistical features are the number of inbound packets with a finish flag and the mean of inbound packet sizes. Examples of many behavioural features are the measure of the connection under question, the number of mutual flows between client and server, both before and after a connection starts or finishes. Examples of many behavioural features are the measure of the connection under question, the number of mutual flows between client and server, both before and after a connection starts or finishes. In order to generalise the finding,

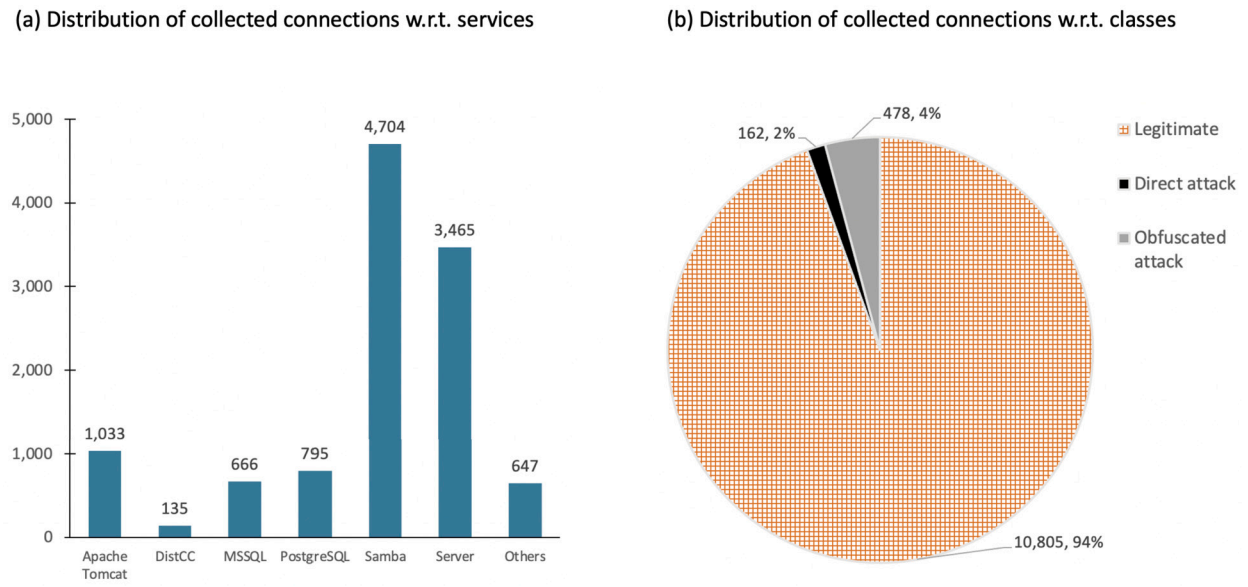

Fig. 3. Distribution of collected connections w.r.t. (a) services and (b) classes.

an additional experiment on another benchmark dataset published in the area of adversarial training for botnet IDS, Drelab [5], is conducted and discussed in Section 4.5.

4.2. Experimental setup

To achieve a rigorous performance assessment of the proposed framework, referred to as ABC-MultiClus hereafter for conciseness, a collection of relevant and state-of-the-art methods is explored. These include the underlying baseline of ABC-MultiClus itself: FFS (forward feature selection), which is the original work on this specific dataset [3]; SingleClus (single-clustering based system) [15]; and MultiClus (multiple-clustering based system) [7] that employs an overall fitness function. Furthermore, three benchmark undersampling methods are also evaluated, namely, RUS [16] and its ensemble-based extensions: RUSBoost [16] and IRUS [17]. System parameters involved are set in accordance with the common practice in the literature [41], with the desired sizes of majority and minority classes fixed to be the same after an undersampling procedure.

In addition, three alternative works are also compared: MPU [18], DBIG-US [19] and CBIS [20]. Particularly, DBIG-US [19] utilises DBSCAN clustering to firstly remove outliers or those on the border line, and then a graph-based elimination is taken to exclude majority-class instances. CBIS [20] firstly exploits the clustering technique of AP (affinity propagation) to obtain groups of samples belonging to the majority class. Then, group-specific representatives are identified using the instance selection method of IB3. To complete the picture, the effectiveness of an oversampling technique is also discussed as it serves as the basis upon which to increase the size of T_1 such that $|T_1| = |T_0|$. For this, a recent variation of SMOTE known as Outlier-SMOTE is examined, using the default parameter setting as per the original report [21]. Further experimental settings are summarised below.

- For ABC-MultiClus and its baseline MultiClus, $M = 150$ is set for the size of ensemble clustering, and the same generation strategies given earlier are exploited for both.
- For ABC-MultiClus, following common practice for ABS modelling, specific algorithmic parameters are set as follows: the number of food sources $fn = 30$, the maximum number of iterations $maxCyc = 1,000$, and the upper bound of no-improvement generations $lmt = 20$. Initially, the weighting factor κ is set to 0.5 such that the resulting model can be directly compared with MultiClus.
- Following previous works [3,7] that are of a focus on this benchmark dataset, the following classification algorithms are employed to implement ω, developing a classifier from a balance training set: (i) Gaussian-kernel NB, (ii) Radial kernel SVM, (iii) LR, and (iv) C4.5 (a popular DT algorithm with the maximum depth of 10). Another two techniques commonly used in the NIDS research are also included to ensure their discrepancies: (v) RF with the same maximum depth as C4.5 and the size of 100 ensemble members, and (vi) KNN using the number of nearest neighbours of 3.
- Given the quest of delivering a predictive model that remains accurate for those obfuscated attacks, each classifier is trained with connections representing direct attacks and legitimate ones only. Let $T = T_{known} \cup T_{unknown}$ where obfuscated attacks belong to $T_{unknown}$ and others to T_{known}. Particularly, $|T_{unknown}| = 478$ and $|T_{known}| = 10,967$. Note that the imbalance ratio for T_{known} is 66.6, i.e., $\frac{10,805}{162}$.
- Stratified 10-fold cross validation is firstly applied to T_{known} to assess predictive performance of different methods for detecting direct attacks. Note that, for each of these 10 folds, a training set consists of around 9,724 legitimate and 146 direct attack

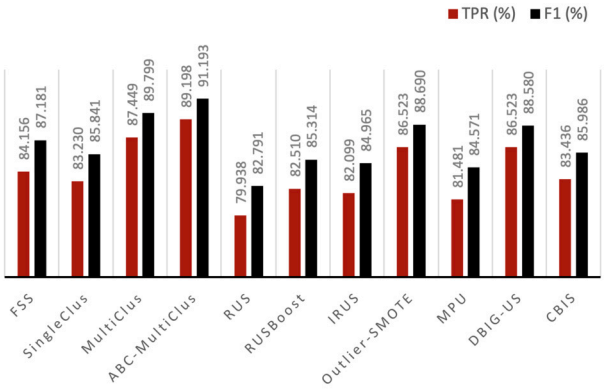

Fig. 4. TPR and F1 scores achieved by investigated techniques as averages across six classifier types and 30 runs of 10-fold cross validation.

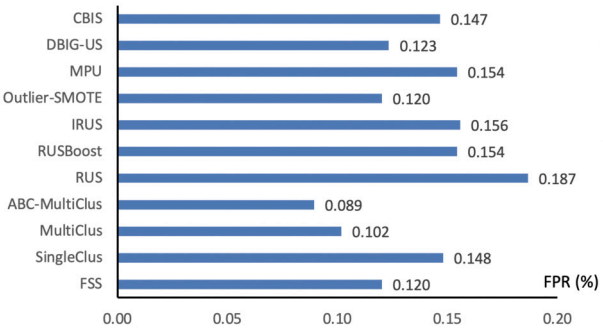

Fig. 5. FPR measures acquired by investigated methods, as averages across classifier types and 30 runs of 10-fold cross validation.

samples, with a test set having around 1,081 legitimate connections and 16 direct attacks. Like the original data, the imbalance ratio of 66.6 is maintained in this process of cross validation. Building from that, the classifier trained in each fold is deployed to categorise 478 samples in $T_{unknown}$ into legitimate or attack types, with the latter classification outcomes being deemed correct. This setting allows a direct comparison of detection performance levels each of the methods achieves for known and adversarial attacks. The assessment is repeated for 30 trials to acquire the average as a reliable conclusion. Conventional metrics of F1, TRP (True Positive Rate) and FPR (False Positive Rate) are exploited to deliver the interpretation of each classifier's performance. Of course, only TPR is reported for the experiment of classifying those unknown instances as all test instances belong to one class. After the report of basic results in Section 4.3, Section 4.4 will provide further detail and discussion on the use of proposed methods with imbalance ratios other than 1. This is to investigate a possible impact of information loss through the undersampling procedure, which has been suggested by [8].

4.3. Experimental results

4.3.1. Detection of direct attacks

Kicking off with a less difficult problem of the two, the results based on the first experiment of 10-fold cross validation on T_{known} are summarised in Figs. 4 and 5, which provide TPR/F1 and FPR scores achieved by all eleven methods. These scores are averages across classification techniques and 30 runs of the aforementioned assessment method, respectively.

According to Fig. 4, the new undersampling framework of ABC-MultiClus makes an improvement to both TPR and F1 scores over all its baseline models. In particular, it reaches 91.193% for the F1 measure, while MultiClus, SingleClus and FFS receive 89.799%, 85.841% and 87.181%, respectively. This underlines the superiority of multiple clustering approach to the single clustering counterpart as a choice to reduce the majority class. This is evident because SingleClus encounters a negative effect of information loss since its F1 is lower than that of FFS. Nonetheless, SingleClus remains effective as compared to the benchmark alternatives such as RUS and the ensemble-based extensions of IRUS and RUSBoost (whose F1 scores are 82.791%, 84.965% and 85.314%). Among the three state-of-the-art methods, DBIG-US, CBIS and MPU, the first two appear to be more accurate than the baseline of SingleClus, while the other only has 84.571% as its F1 score. Moving to the oversampling category, Outlier-SMOTE reaches a high F1 measure of 88.69%, the third highest F1 across eleven recordings. A similar trend is witnessed with those TPR scores across, as shown in Figs. 4, where ABC-MultiClus and RUS are at the top and the bottom end with regards to this performance index (of a TPR measures of 89.198% and 79.938% respectively).

In accordance with the FPR scores illustrated in Fig. 5, the proposed framework is comparable to its baseline of multiple-clustering approach, with measures of 0.089% and 0.102%. In addition, Outlier-SMOTE is competitive in this regard as it obtains the rate of

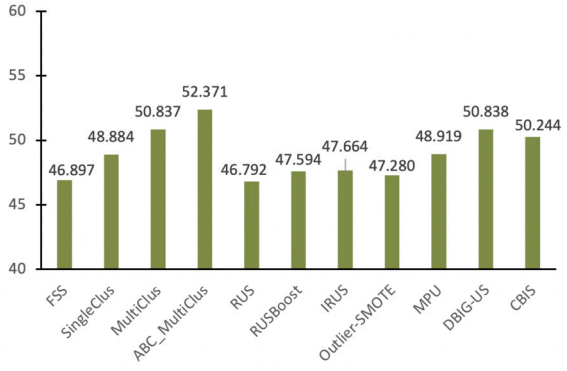

Fig. 6. Average TPR scores achieved by investigated methods for classification of obfuscated attacks over 30 runs of 10-fold cross validation.

0.12%. In spite of this interesting observation, an oversampling alternative may lead to the issue of overfitting, especially for unseen cases, which shall be illustrated in the next section. For other undersampling models, the best and the worst scores of 0.123% and 0.187% are observed with DBIG-US and RUS. With these results, ABC-MultiClus is shown to be able to support training of an accurate classifier for known attacks, based on the feature subset identified by FFS, the original work.

4.3.2. Classification of obfuscated attacks

Fig. 6 presents TPR scores obtained by the eleven techniques investigated for classification of 478 obfuscated samples (either legitimate or attack ones), which have been deliberately excluded from the training processes, via 10-fold cross validation. Concluded from 30 trials of this evaluation, good scores expected would be higher than 46.897% which is observed with the baseline model of FFS. The first observation from Fig. 6 is that the clustering directed approach is more effective than conventional models of undersampling, with SingleClus receiving a score of 48.884% whilst RUS, RUSBoost and IRUS having scores of 46.792%, 47.594% and 47.664%, respectively. Moreover, multiple-clustering based approaches, namely ABC-MultiClus and MultiClus, offer two of the most accurate results with the scores of 52.371% and 50.837%, while the recently invented DBIG-US and CBIS appear to be comparable to MultiClus. Unlike the prediction result of those known samples, Outlier-SMOTE becomes less accurate for the collection of obfuscated attacks, as it achieves a TPR score of 47.28%. This finding shows that the learned model suffers from overfitting, not sufficiently generalised to new patterns.

Table 2 gives further details of averaged TPRs and corresponding standard deviations, which are categorised by a match of classifier types and investigated methods. As can be seen, the proposed undersampling framework generally outperforms the basic multiple-clustering counterpart and other compared methods. The highest TPR rate of 86.192% is witnessed with the coupling of ABC-MultiClus and NB classifier. That is, with clustering-based undersampling framework, a classifier is able to learn more from the minority class, thereby generalising to new attack patterns, especially when integrated with ABC-based selection of representative samples. It is interesting to see a few classifiers that perform well with known attacks become less accurate for adversarial samples. Specific to FFS and ABC-MultiClus, Fig. 7 summarises the average drops in TPR rate across the six classifiers. The biggest drops of around 65% and 58% are observed with SVM and C4.5, while LR, NB and KNN seem to be more robust in this respect. By switching from the radial kernel to a linear one, the TPR rate of SVM with FFS improves from 15.69% to 21.351%, while that of SVM with ABC-MultiClus increases from 21.548% to 24.578%. These imply a likely problem of overfitting a model, i.e., support vectors, to (i) best separate known attacks and those legitimate samples, and (ii) project original features to a kernel-led space, which might not be robust to changes in a feature vector. For C4.5 with one fixed tree structure, a mutation of feature value can lead to a different decision branch, thus complicating the prediction outcome. It might be better working with multiple trees, some of which are not affected by an adversarial signature. As shown in Table 2, the TPR rates of RF are usually higher than those of C4.5.

4.3.3. Statistic significance analysis

Whilst averages across trials provide a good indication of the performance of the proposed approach in action, confidence-led assessment will offer further assurance regarding the efficacy of any implemented system [7,21]. Following [9,42], one method $b \in \Psi$ is statistically deemed to be 'significantly better' than any other $\forall c \in \Psi, c \neq b$, if the performance indices concerned are of a confidence level of at least 95%. Here, Ψ denotes the set of eleven methods under comparative investigation.

Let $\mu_b^t(\omega)$ be the average of TPR scores obtained by the method $b \in \Psi$ from the t-th trial of ζ-fold cross validation using a classifier ω, where ζ is 10 and $\omega \in \{NB, C4.5, SVM, LR, RF, KNN\}$ in this work. Instead of exploiting $\mu_b^t(\omega)$ for a direct comparison, the 95% confidence interval $[L(\mu_b^t(\omega)), U(\mu_b^t(\omega))]$ is put in place with lower $L(\mu_b^t(\omega))$ and upper $U(\mu_b^t(\omega))$ bounds being specified by the following, where $SD_b^t(\omega)$ represents a standard deviation of $\mu_b^t(\omega)$:

$$L(\mu_b^t(\omega)) = mu_b^t(\omega) - 1.96\frac{SD_b^t(\omega)}{\sqrt{\zeta}} \tag{6}$$

$$U(\mu_b^t(\omega)) = mu_b^t(\omega) + 1.96\frac{SD_b^t(\omega)}{\sqrt{\zeta}} \tag{7}$$

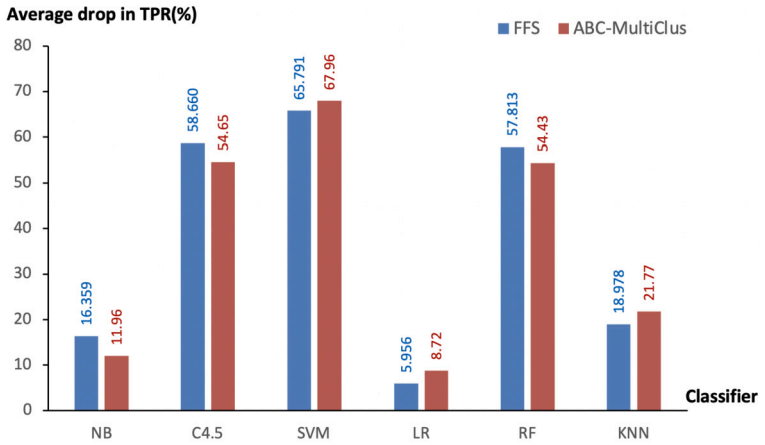

Fig. 7. A comparison of average drop in TPR scores achieved by different classifiers, i.e., the difference between TPR scores for the prediction of known and adversarial attacks. These focus on details of FFS and the proposed method.

Table 2
Average TPR scores specific to pairs of investigated method and classifier, summarised across 30 runs of 10-fold cross validation with corresponding standard deviations provided in (brackets).

Investigated Method	NB	C4.5	SVM	LR	RF	KNN
FFS	81.172 (1.282)	36.402 (1.191)	15.690 (1.246)	63.180 (1.352)	37.866 (1.417)	47.071 (1.214)
SingleClus	82.636 (1.776)	38.494 (1.784)	17.782 (1.603)	65.272 (1.548)	39.540 (1.605)	49.582 (1.432)
MultiClus	85.146 (1.904)	40.377 (1.881)	19.665 (1.904)	67.782 (1.810)	41.004 (1.918)	51.046 (1.563)
ABC-MultiClus	86.192 (1.802)	42.259 (1.976)	21.548 (2.017)	70.293 (1.982)	43.096 (2.003)	52.301 (1.611)
RUS	80.753 (2.622)	35.983 (2.721)	16.946 (2.673)	63.808 (2.514)	36.192 (2.654)	47.071 (1.756)
RUSBoost	81.172 (2.219)	36.402 (2.016)	17.573 (2.562)	64.644 (2.273)	37.866 (2.356)	47.908 (1.809)
IRUS	81.381 (2.134)	36.611 (2.073)	17.782 (2.123)	64.435 (2.281)	38.075 (2.211)	47.699 (1.835)
Outlier-SMOTE	79.498 (2.227)	36.192 (2.160)	16.946 (2.262)	65.272 (2.185)	36.611 (2.098)	49.163 (2.044)
MPU	82.218 (1.602)	38.912 (1.654)	17.992 (1.896)	65.063 (1.712)	40.377 (1.579)	48.954 (1.403)
DBIG-US	85.356 (2.102)	40.167 (2.315)	19.665 (2.208)	67.364 (2.017)	41.213 (1.987)	51.255 (1.927)
CBIS	84.310 (1.821)	39.331 (1.425)	18.828 (1.679)	66.109 (1.791)	41.004 (1.693)	51.883 (1.784)

From the above, it can be decided whether $\mu_b^t(\omega)$ is higher than $\mu_c^t(\omega)$, such that the method b performs better than c in the t-th run only when $L(\mu_b^t(\omega)) > U(\mu_c^t(\omega))$. Formally, a function $better(b, c, \omega, t)$ returns 1 if the previous condition is true using the classifier ω, and 0 otherwise. The opposite conclusion of b being worse than c in the same trial and classifier can be similarly derived for the case where $U(\mu_b^t(\omega)) < L(\mu_c^t(\omega))$, namely, $worse(b, c, \omega, t) = 1$. Then, the number of times $B(b)$ that a technique $b \in \Psi$ is significantly better than others can be estimated as follows.

$$B(b) = \forall_\omega \forall_{c \in \Psi, c \neq b} \forall_{t=1\ldots30} \;\; better(b, c, \omega, t) \tag{8}$$

Likewise, the frequency $W(b)$ that $b \in \Psi$ is significantly worse than others can be defined by

$$W(b) = \forall_\omega \forall_{c \in \Psi, c \neq b} \forall_{t=1\ldots30} \;\; worse(b, c, \omega, t) \tag{9}$$

Provided these, the summarisation of both significant comparisons is determined as $B(b) - W(b)$, with a positive value signifying the method b is often more accurate than others, and less accurate otherwise. Table 3 shows B-W statistics of all eleven methods investigated in this study. Clearly, ABC-MultiClus is more accurate than others, whilst Outlier-SMOTE and RUS are at the other end of this spectrum. To reinforce this observation, Fig. 8 presents both better and worse frequencies of those top 6 techniques given in the previous table, including SingleClus, MultiClus, ABC-MultiClus, MPU, DBIG-US and CBIS, respectively. This is in line with the previous finding in that ABC-MultiClus is the most effective among these undersampling alternatives, with SingleClus appearing to be the least accurate.

4.4. Parameter analysis and implications

This section investigates the effects different parameter settings of ABC-MultiClus upon its classification, especially on the implication of using it as a system to handle highly imbalanced data.

Table 3
B-W statistics for investigated methods, categorised with respect to classifier types.

Method	NB	C4.5	SVM	LR	RF	KNN	Total
FFS	-18	-19	-19	-31	-19	-31	-137
SingleClus	2	5	-9	1	3	1	3
MultiClus	26	25	19	27	27	27	151
ABC-MultiClus	51	50	39	49	49	48	286
RUS	-35	-37	-20	-33	-37	-33	-195
RUSBoost	-14	-27	-9	-14	-27	-14	-105
IRUS	-14	-19	-6	-19	-19	-19	-96
Outlier-SMOTE	-48	-31	-21	-6	-31	-6	-143
MPU	4	4	-1	-7	6	-5	1
DBIG-US	27	28	15	21	28	21	140
CBIS	17	19	11	11	19	11	88

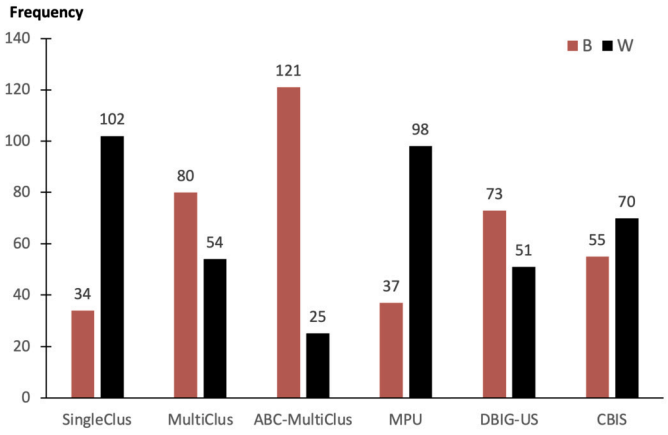

Fig. 8. Statistics of better and worse results achieved by comparing only the top six methods shown in Table 3.

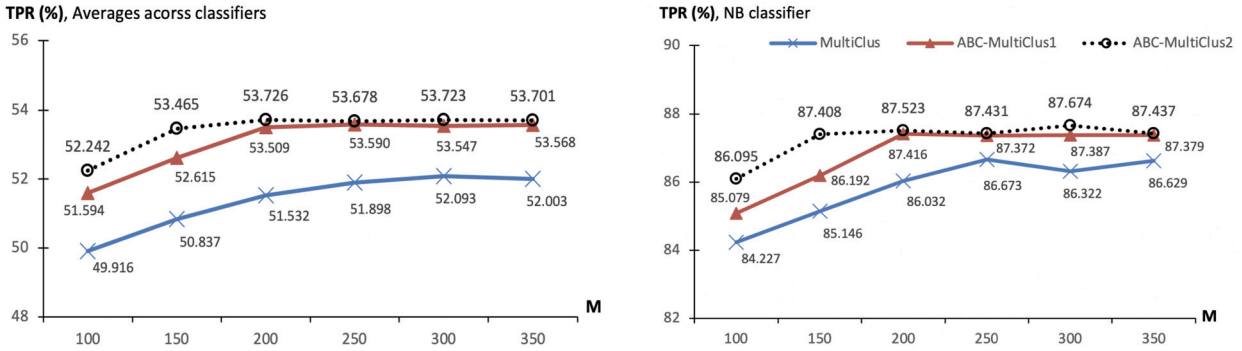

Fig. 9. Average TPR scores obtained by ABC-MultiClus and MultiClus using different $M \in \{100, 150, 200, 250, 300, 350\}$, across all classifier types investigated (left) and results returned by NB (right), where ABC-MultiClus1 and ABC-MultiClus2 respectively employ the random and global-best selection of food source for onlooker bees.

4.4.1. Impact of ensemble size and computational complexity

The first parameter that may influence the quality of the proposed approach is the size of ensemble clustering (which has been set to $M = 150$ so far). To reveal any potential association between the size and classification performance, the experiment on classifying obfuscated attacks is repeated herein using $M \in \{100, 150, 200, 250, 300, 350\}$. Fig. 9 illustrates the average TPR scores acquired by ABC-MultiClus and MultiClus, with those results across four classifiers depicted on the left and those specific to NB on the right. Two variations of ABC-MultiClus (ABC-MultiClus1 and ABC-MultiClus2 that employ the random and global-best selection of food source for onlooker bees, respectively) are examined. This is not only to generalise the present finding, but also to observe the effects that different exploiting strategies employed by ABC may have.

The results indicate that a bigger ensemble size usually leads to a higher TPR score for all three models under examination, with ABC-MultiClus1 reaching peaks approximately at $M = 200$ and above. Specific to ABC-MultiClus1 and NB, the score of 87.416% is attained as M inclines from 150 to 200, where a similar tendency is also observed in the average across of all classifiers. However,

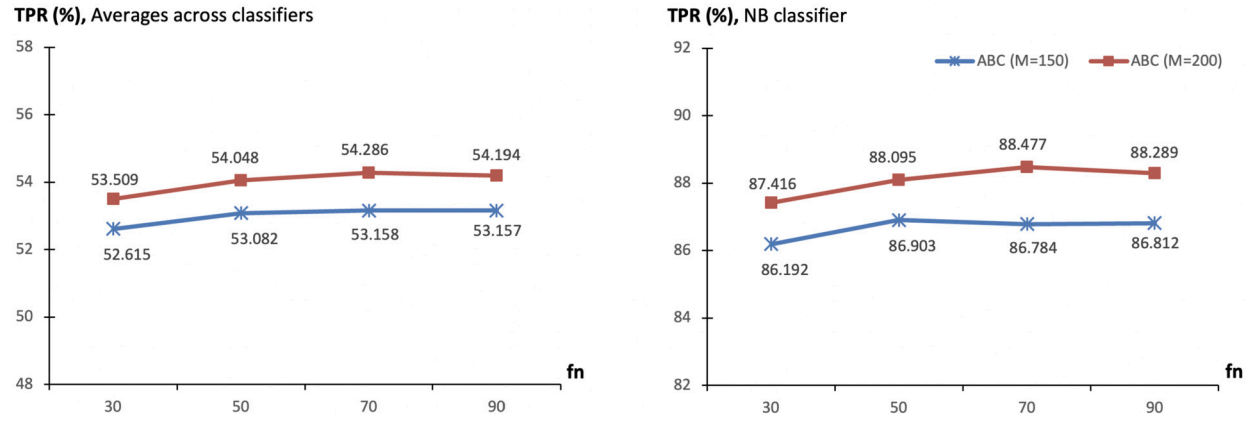

Fig. 10. TPR scores obtained by two variations of ABC-MultiClus ($M = 150$ and $M = 200$) using different $fn \in \{30, 50, 70, 90\}$, presented as averages across classifier types (left) and those specific to NB (right).

enlarging an ensemble beyond 200, hence the number of centroids in a pool, may not yield further improvement, but inevitably magnify the complexity. Similar results to ABC-MultiClus1 are also witnessed for ABC-MultiClus2, with the latter climbs up to peaks more rapidly than the other. Indeed, ABC-MultiClus2 may obtain the highest scores with a smaller M of 150, where those onlooker bees are able to find better food sources using the global-best heuristic (than the original choice of random operation). It is also important to note that an iterative greedy search employed by MultiClus often requires a bigger pool to be in par with the other two.

These results also present a hint that the convergence of ABC-MultiClus may be accelerated with more empirical investigations given a certain real-world application. As an initial consideration, computational complexity of the multiple-clustering based systems is discussed here. At first, the complexity of generating an ensemble of M members using k-means is $O(nkM)$, with n and k being the number of training samples and the average cluster number. The complexity of MultiClus is the combination of the cost due to this generation and that due to a greedy search for $\chi, \chi = |T_1|$ (the number of representative centroids), i.e., $O(nkM + \chi^2)$. Provided that χ is likely to be much smaller than n (which normally is), this reduces to $O(nkM)$, with M being a major impact on resource requirement. For ABC-MultiClus, the second component incurs a higher complexity because in this case, it becomes $O(maxCyc.fn.\chi)$, with $maxCyc$ and fn denoting the maximum iterations of ABC search and the number of food sources, respectively. It is expected that ABC-MultiClus is more expensive than its predecessor since $maxCyc$ should be sufficiently high to ensure convergence over the global optimisation process. This is also the case for fn that corresponds to number of bees working in a colony [11].

Similar to the investigation of the parameter M, experiments are repeated with $fn \in \{30, 50, 70, 90\}$, with the results illustrated in Fig. 10. It can be seen that a larger fn typically leads to an improvement for both variants of ABC-MultiClus, using $M = 150$ and $M = 200$. With the default value of $maxCyc = 1,000$, $fn > 70$ may not lead to further increase in the TPR scores. Nonetheless, the complexity may reduce if the number of maximum iterations is lowered.

4.4.2. Impact of weighting factor

Another important parameter of running ABC-MultiClus is the weighting factor $\kappa \in [0, 1]$ defined in the fitness function Ω^*, which integrates two different criteria for assessing a given representative centroid. On one hand, the diversity among representatives themselves contributes more as $\kappa > 0.5$. On the other, the difference between these centroids and instances of the minority class becomes more influential. As such, the experiment is repeated again for ABC-MultiClus ($M = 200$ and $fn = 70$ are set in accordance with the results shown in Fig. 10) using different $\kappa \in \{0, 1, 0.2, \ldots, 0.9\}$. For the task of classifying obfuscated connections, Fig. 11 summarises average TPR scores across classifier types on the left, and those specific to NB on the right. In both cases, the TPR measures start increasing as κ becomes larger than 0.3, with the best values being seen at κ around 0.8-0.9.

Fig. 12 also provides further details on the effect that κ has on the performance over the two tasks of classifying basic legitimate-direct attack and obfuscated connections. They indicate the different behavioural patterns exhibited by ABC-MultiClus, where a low value of κ ($\kappa = 0.3$) leads to the best performance for the first task (91.146%) and a high value ($\kappa = 0.8$) for the other (55.21%). To further examine the proposed approach through experimental evaluations, Figs. 13 and 14 are given to illustrate the relation between κ, spaces of likely selected centroid representatives and obfuscated instances. In the former, obfuscated connections that are mutated instances of known direct attacks seem to be most likely to appear in the overlapping region between the two classes, noticing the area encapsulated by the dashed red circle. Given a low κ value, ABC-MultiClus prefers centroids that are different from direct-attack samples, which can be visualised by the area within the dashed blue circle. This effect of moving away from the overlapping region and the borderline between classes becomes clearer as κ reduces its value. As shown in Fig. 12, such a setting suits a basic classification of legitimate and direct-attack connections, but not for obfuscated ones. To allow a classifier to learn more from mutated patterns, the blue circle should move closer to the red. This is depicted in Fig. 14, where those centroids within the red circle have a better chance to be selected by ABC-MultiClus and hence, be learned by the target classifier.

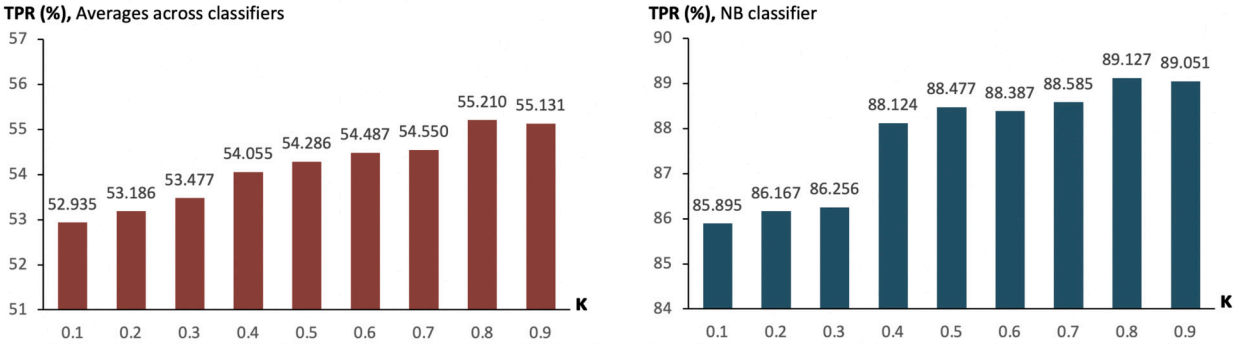

Fig. 11. Average TPR scores obtained by ABC-MultiClus ($M = 200$ and $fn = 70$) using different $\kappa \in \{0, 1, 0.2, \ldots, 0.9\}$ across classifier types (left) and those specific to NB (right).

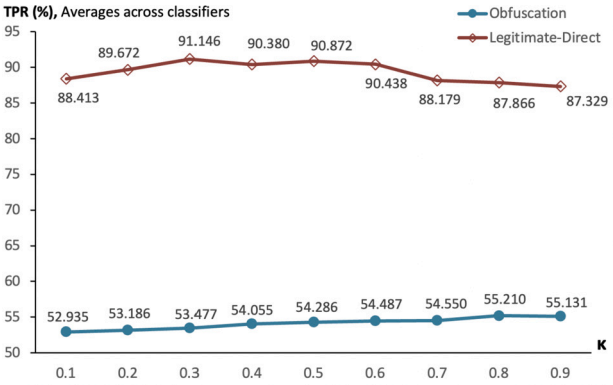

Fig. 12. TPR scores obtained by ABC-MultiClus ($M = 200$ and $fn = 70$) as averages across classifier types with respect to $\kappa \in \{0, 1, 0.2, \ldots, 0.9\}$, presented for two tasks of classifying basic legitimate-direct attack and obfuscated connections.

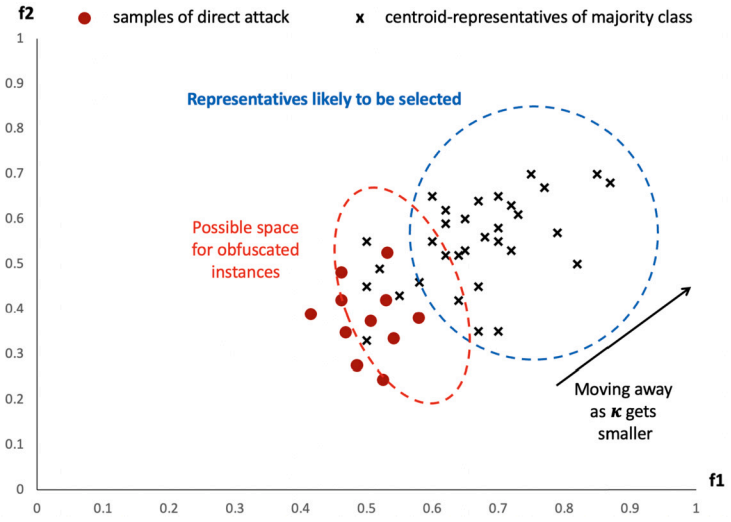

Fig. 13. Illustration of relation between **low** κ, spaces of likely selected centroid representatives and obfuscated instances.

4.4.3. Impact of legitimate-to-attack ratio & implication of greatly imbalance data

For the results presented and discussed thus far, the default legitimate-to-attack ratio in a training set created by ABC-MultiClus is 1:1, thus needing an additional investigation on a possible loss of informative samples from the majority class [8]. To achieve this, the previous experiment is repeated for another two ratios of 10:1 and 20:1. The results shown in Fig. 15 are summarised from 30 trials using the default setting of $M = 150$, $fn = 30$ and $\kappa = 0.5$. With the TPR rates specific to FFS, there is a sign of losing some of legitimate signatures that support the differentiation between adversarial attacks and normal connections. In particular to NB, the

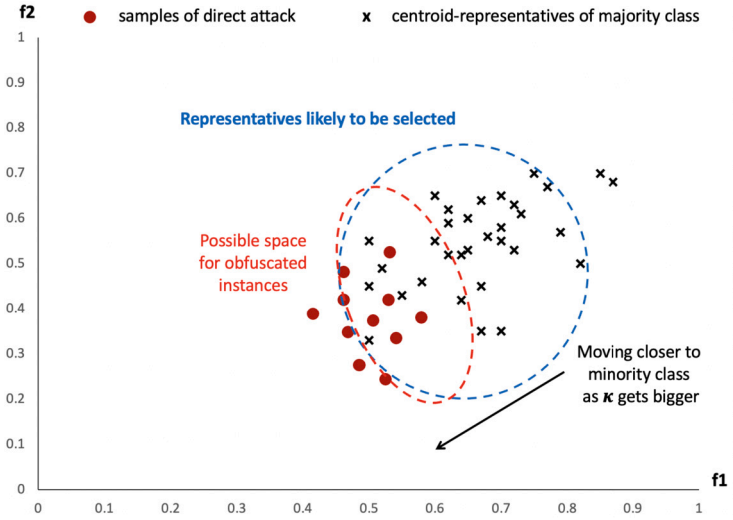

Fig. 14. Illustration of relation between **high** κ, spaces of likely selected centroid representatives and obfuscated instances.

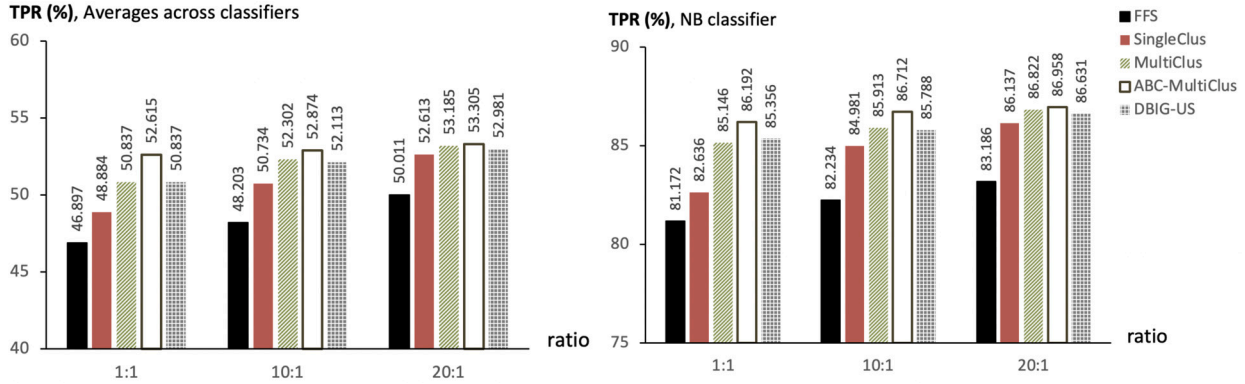

Fig. 15. Average TPR scores obtained by ABC-MultiClus ($M = 150$, $fn = 30$ and $\kappa = 0.5$) for three different legitimate-to-attack ratios in a target training set: 1:1, 10:1 and 20:1, respectively, across classifier types (left) and those specific to NB (right).

TPR rates increases from 81.172% to 83.186% with the ratios 1:1 and 20:1, respectively. The two multiple clustering-based methods sustain this loss rather well up to the ratio of 10:1 but fail at the bigger size of legitimate samples. To be exact, ABC-MultiClus obtains the TPR rates of 86.192% at the 1:1 ratio and 86.958% at the ratio of 20:1, while these are 85.146% and 86.822% for MultiClus. As more legitimate samples are available, SingleClus becomes more effective and comparable to ABC-MultiClus, MultiClus and DBIG-US, especially at the 20:1 ratio. Similar trends are also observed with the averages across classifier types. With these results, the default ratio of 1:1 exploited by the proposed method may be sub-optimal in terms of providing the most accurate prediction, despite making the resulting training process more efficient with a smaller set of legitimate connections.

The next issue investigated here is the implication of ABC-MultiClus (with the default legitimate-to-attack ratio in a training set of 1:1) on greatly imbalance class distribution, which is frequently encountered in real-world problems like fraud detection, cancer diagnosis and discovery science, among many others. For all previous evaluations, the imbalance ratio is constant at 66.6, given that there are 10,805 legitimate and 162 direct-attack connections. With those attack samples being randomly dropped from the original data set, three variations containing only 122, 81 and 41 attack samples are created to exhibit higher imbalance ratios of 88.5, 133.4 and 263.5, respectively. From this, the basic comparative experimental studies carried out earlier are repeated again for applying ABC-MultiClus to the aforementioned newly created datasets, using the default setting of $M = 150$, $fn = 30$ and $\kappa = 0.5$. Based on 30 trials, Fig. 16 presents average TPR scores of classifying obfuscated connections across classifier types on the left, and those specific to NB on the right. It confirms the assumption that classification performance would decline as the ratio concerned becomes larger (e.g., the TPR scores of ABC-MultiClus with NB fall from 86.192% to 78.014% as the ratio increases from 66.6 to 263.5). This observation is common to all five undersampling methods examined here, but fortunately ABC-MultiClus and DBIG-US are normally more robust than the rest. However, as the ratio reaches a very high value, the difference among these becomes only marginal with FFS being comparable to MultiClus and SingleClus.

To generalise the above finding, a further study is conducted with another data collection that possesses the imbalance ratio higher than 300. This follows the recent literature [43], by employing two datasets (named Data1 and Data2) that have been exploited to

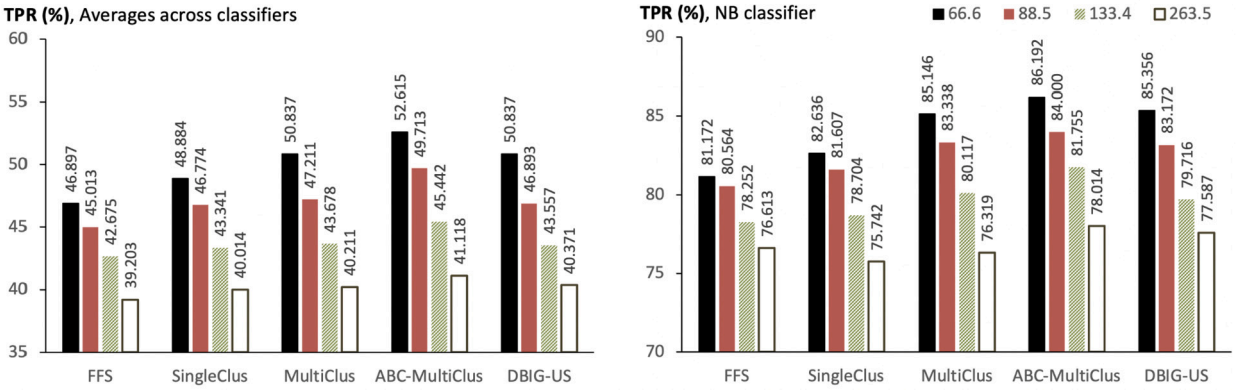

Fig. 16. Average TPR scores obtained by ABC-MultiClus ($M = 150$, $fn = 30$ and $\kappa = 0.5$) on four dataset variations, with the imbalance ratios of 66.6, 88.5, 133.4 and 263.5, respectively, across classifier types (left) and those specific to NB (right).

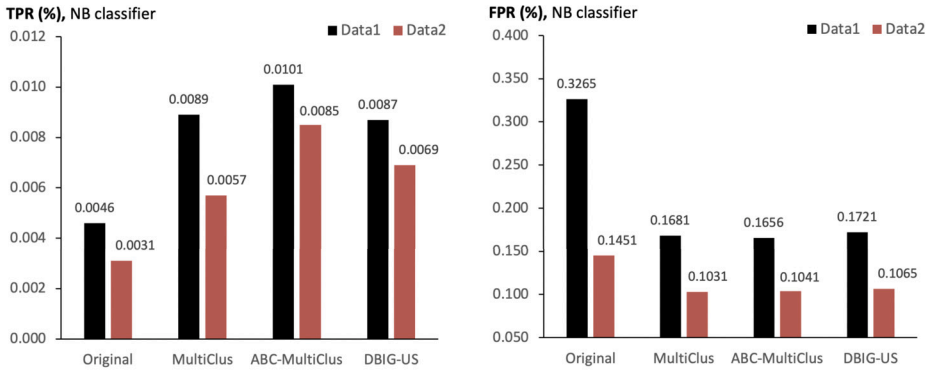

Fig. 17. TPR and FPR scores obtained by ABC-MultiClus ($M = 150$, $fn = 30$ and $\kappa = 0.5$) and alternative undersampling methods, using NB classifier on datasets Data1 and Data2.

evaluate performance of ML models for transient event detection. Both datasets address binary classification problems, where Class-1 is the minority and presents real transients, while bogus instances caused by errors in equipment and image alignment belong to Class-0. In particular, for Data1, there are 5,973 and 16 samples of Class-0 and Class-1, with the imbalance ratio being 373.313. A slightly higher ratio of 375.167 is seen in Data2 that contains 6,573 and 18 instances of those classes. To ease comparison with the existing work, NB is employed to deliver both TPR and FPR scores acquired by ABC-MultiClus and other undersampling companions. Fig. 17 provides average scores across 30 trials of cross validation for both data sets, with Original corresponding to the classification outcome from the original work compared. From these results it is clear that ABC-MultiClus still proves to be an effective choice of undersampling method, even with such hugely large imbalance ratios.

4.5. Application to adversarial botnet attacks against NIDS

Based on the recent survey of [44], adversarial attacks to NIDS, especially the evasive type, have gained a great deal of attention as a direct consequent of similar research in computer vision. In particular, three main categories of defences have been identified: parameter protection, adversarial detection, and robustness optimisation, respectively. The first family attempts to hide algorithmic variables and training data used to develop a detection model, while the second deploys a statistical tool or a ML-based secondary model to identify the possible mutated signatures exhibited by incoming traffics. The last category focuses on making a detector robust to unseen attack samples. This can be accomplished by either improving the data preprocessing stage such that the model is more generalised based on known attacks, fine tuning the model with other datasets defined by the same feature space (however, with a chance of poisoning attacks [6]), or training the model with adversarial examples. The proposed method belongs to the first sub-category of this group as it optimises the robustness of classifiers through the handling of imbalanced classes.

Despite good results reported in the previous sections, a further investigation with other datasets developed in this field is required to generalise findings. For the current work, a dataset created for adversarial training of botnet IDS may well be employed here, see [45] for an example of the recent development. Thanks to the research team of Drelab [5] that adopts two deep reinforcement-learning or DRL methods to produce botnet adversarial attacks: Double Deep Q-Network and Deep Sarsa, which will be referred to shortly as DDQN and SARSA hereafter. Table 4 summarises details of the eight datasets, i.e., subsets of the original Drelab, to be employed in the next experiment. To simplify this empirical study, two public datasets of CTU and BOTNET are included to provide benign signatures where malicious samples are drawn from network flows belonging to two botnet families, Menti and Rbot. A classification

Table 4
Details of the botnet adversarial-attack datasets that are extracted from the Drelab collection.

(No.) Dataset	Training: no. of benign samples	Training: no. of malicious samples	Training: imbalance ratio	Test: no. of adversarial attacks
(1) CTU-Menti-DDQN	2,582,434	2,825	914.136	2,825
(2) CTU-Menti-SARSA	2,582,434	2,825	914.136	2,823
(3) CTU-Rbot-DDQN	2,582,434	27,456	94.057	27,452
(4) CTU-Rbot-SARSA	2,582,434	27,456	94.057	27,425
(5) BOTNET-Menti-DDQN	73,721	7,923	9.305	7,754
(6) BOTNET-Menti-SARSA	73,721	7,923	9.305	7,920
(7) BOTNET-Rbot-DDQN	73,721	29,602	2.490	29,503
(8) BOTNET-Rbot-SARSA	73,721	29,602	2.490	28,267

Table 5
Average TPR scores across 30 trials (with corresponding standard deviation values given in brackets) obtained by three undersampling methods (using the legitimate-to-attack ratio of 1:1) and Original that is the baseline of leaving the original data distribution untouched. These are presented with respect to those eight datasets shown in Table 4 and four classification techniques, which appear in the previous experiment to be robust to obfuscated or mutated signatures.

Dataset No.	Method	RF	NB	KNN	LR
1	Original	0.942 (0.321)	93.027 (0.000)	0.814 (0.000)	0.000 (0.000)
	ABC-MultiClus	97.944 (1.674)	93.026 (0.009)	6.156 (0.520)	34.886 (0.953)
	MultiClus	95.785 (1.934)	93.027 (0.015)	5.249 (0.714)	31.379 (0.883)
	DBIG-US	96.412 (1.307)	93.026 (0.012)	4.883 (0.605)	32.964 (0.724)
2	Original	0.716 (0.282)	93.163 (0.000)	56.961 (0.000)	0.000 (0.000)
	ABC-MultiClus	98.406 (0.651)	93.166 (0.009)	92.812 (1.446)	94.158 (0.057)
	MultiClus	96.732 (1.004)	93.154 (0.023)	91.045 (1.652)	92.677 (0.079)
	DBIG-US	95.137 (0.983)	93.163 (0.008)	91.594 (1.101)	91.205 (0.096)
3	Original	0.051 (0.035)	99.580 (0.000)	2.455 (0.000)	92.412 (0.000)
	ABC-MultiClus	78.836 (2.020)	99.580 (0.000)	92.538 (1.877)	99.724 (0.037)
	MultiClus	77.047 (1.924)	99.580 (0.000)	90.212 (1.924)	96.451 (0.088)
	DBIG-US	75.579 (1.756)	99.580 (0.000)	92.404 (2.085)	97.338 (0.044)
4	Original	0.971 (0.189)	98.818 (0.000)	3.030 (0.000)	92.598 (0.000)
	ABC-MultiClus	79.160 (1.726)	98.892 (0.004)	93.812 (1.770)	99.086 (0.331)
	MultiClus	76.908 (1.671)	98.809 (0.006)	92.312 (1.677)	98.766 (0.634)
	DBIG-US	75.163 (1.812)	98.823 (0.004)	91.784 (1.502)	99.122 (0.431)
5	Original	40.452 (2.373)	95.448 (0.000)	93.977 (0.000)	90.702 (0.000)
	ABC-MultiClus	90.886 (1.493)	95.440 (0.000)	94.892 (0.224)	91.140 (2.130)
	MultiClus	87.483 (1.293)	95.440 (0.000)	93.612 (0.340)	88.751 (2.338)
	DBIG-US	86.117 (1.205)	95.440 (0.000)	92.731 (0.516)	87.433 (1.986)
6	Original	6.288 (2.213)	93.447 (0.000)	72.235 (0.000)	92.348 (0.000)
	ABC-MultiClus	16.442 (1.676)	93.440 (0.000)	96.350 (1.708)	92.592 (0.178)
	MultiClus	15.767 (1.874)	93.440 (0.000)	94.104 (1.509)	92.302 (0.164)
	DBIG-US	14.438 (1.303)	93.440 (0.000)	95.879 (1.617)	92.411 (0.193)
7	Original	99.334 (0.153)	97.631 (0.000)	92.296 (0.000)	83.320 (0.000)
	ABC-MultiClus	99.533 (0.226)	97.638 (0.003)	92.693 (0.440)	83.919 (0.085)
	MultiClus	99.304 (0.214)	97.458 (0.008)	92.456 (0.383)	83.332 (0.103)
	DBIG-US	99.211 (0.315)	97.087 (0.014)	91.231 (0.211)	84.241 (0.092)
8	Original	96.398 (3.422)	94.173 (0.000)	90.561 (0.000)	81.862 (0.000)
	ABC-MultiClus	98.818 (1.297)	94.624 (0.013)	92.044 (0.481)	85.334 (0.667)
	MultiClus	96.210 (1.306)	94.651 (0.034)	90.013 (0.502)	84.127 (0.771)
	DBIG-US	98.114 (1.064)	94.145 (0.027)	90.383 (0.448)	85.212 (0.503)

model will be trained using different combinations of these and then assessed on the unseen sets of adversarial attacks prepared by either using DDQN or SARSA. Please consult [46] for further detail.

Table 5 shows average TPR scores across 30 trials achieved by three undersampling methods (using the legitimate-to-attack ratio of 1:1) and Original that is the baseline of leaving the original data distribution untouched. These are presented with respect to those eight datasets shown in Table 4 and four classification techniques, which appear in previous experiments to be robust to obfuscated or mutated signatures. The results suggest that ABC-MultiClus is usually a better choice of undersampling method than MultiClus and DBIG-US. It is also interesting to see that these techniques significantly improve the detection rates of adversarial attacks in several datasets, see the datasets 1, 4, 6 and 8 for examples. However, a similar improvement made to NB is minimal as it has already been highly accurate using the original data distribution.

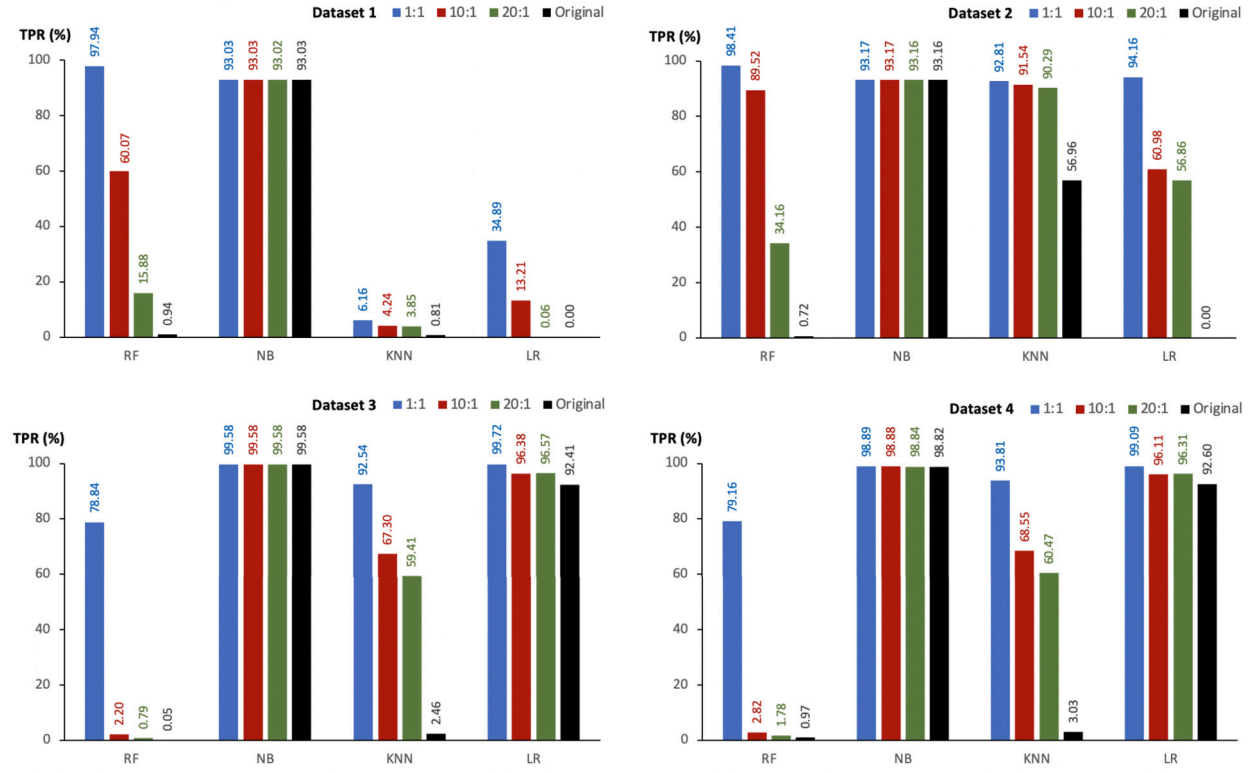

Fig. 18. Average TPR rates across 30 trials obtained by the Original and ABC-MultiClus variations that make use of the legitimate-to-attack ratios of 1:1 (i.e., the default setting), 10:1 and 20:1, respectively.

Similar to the investigation reported in Section 4.4, Fig. 18 summarises average TPR rates across 30 trials obtained by the Original and ABC-MultiClus variations that make use of legitimate-to-attack ratios of 1:1 (i.e., the default setting), 10:1 and 20:1, respectively. With this experiment setting, only Datasets 1 to 4 are included while the others are with the original ratios of less than 10:1. From this illustration, ABC-MultiClus can consistently improve the detection rate of adversarial attacks in these four datasets across the three different ratios. In particular, it is able to sustain the effect of information loss from reducing the ratio down to only 1:1, which is often seen with RF, KNN and LR. For the NB classification method, the performance of ABC-MultiClus using the default ratio of 1:1 is in par with the Original and other ratio alternatives. Despite no obvious improvement in the detection rate, a training of an NB model can be more efficient with a much smaller dataset.

5. Conclusion

This paper has introduced a new undersampling framework to improve predictive performance of ML-based NIDS, especially the classification of obfuscated attacks. As a complement to the original work that focuses on seeking an optimal subset of features, the current research attempts to obtain training data that contains the same number of instances belonging to binary classes. It extends the recent concept of multiple-clustering based undersampling to reduce the size of majority class, thus allowing a target classifier to learn more from the minority. Instead of employing a simple greedy search to select representatives from a pool of centroids acquired from a clustering ensemble, the computational process involved is designed as a global optimisation problem to which the classical ABC algorithm is applied. Particularly, details of the pool of representatives and selected centroids are encoded as a food source with a fitness function to address two different performance criteria concerned. A weighting factor is introduced to combine these criteria in a single fitness assessment, thus allowing the resulting selection method to fit different assumptions on class overlapping.

Systematic comparative results have been achieved using several classifier types and involving different parameter settings. The proposed approach generally outperforms its multiple- and single-clustering baseline counterparts. It has proven as accurate as several state-of-the-art techniques introduced in the literature for handling the class imbalance issue. Whilst promising, the present approach is more expensive than those initial models, making it less efficient for analysing big data or for applications that require online learning. This work opens up interesting avenues for further investigation. First, marrying the proposed work with federated learning may help to resolve the curse of data volume [47]. Second, a comparative study of exploiting different swarm intelligence algorithms [48] and variants of ABC [11] for the selection of representative centroids would be interesting. Hence, effective models might be disclosed and further explored to new domain problems, including the potential employment of cluster-wise neighbour determination procedure to discretise feature domains [49] and to deal with impute missing values [50].

CRediT authorship contribution statement

Tonkla Maneerat: Writing – original draft, Validation, Software, Methodology, Investigation, Data curation. **Natthakan Iam-On:** Writing – original draft, Validation, Investigation, Conceptualization. **Tossapon Boongoen:** Writing – original draft, Validation, Supervision, Methodology, Formal analysis, Conceptualization. **Khwunta Kirimasthong:** Writing – review & editing, Validation, Supervision, Conceptualization. **Nitin Naik:** Writing – review & editing, Validation, Conceptualization. **Longzhi Yang:** Writing – review & editing, Validation, Conceptualization. **Qiang Shen:** Writing – review & editing, Validation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This research work has been supported by Postgraduate Studentship of MFU, and a collaboration between MFU, Aberystwyth, Northumbria and Aston Universities. It is also partly supported by UK FCDO grant: Research and Innovation for Development in ASEAN (RIDA 2023-24: RSA-03160). For this joint project between Aberystwyth and MFU, the proposed method has been successfully applied to improved burnt scar detection in satellite imaging.

References

[1] N. Naik, R. Diao, Q. Shen, Dynamic fuzzy rule interpolation and its application to intrusion detection, IEEE Trans. Fuzzy Syst. 26 (4) (2018) 1878–1892.
[2] W. Liu, X. Xu, L. Wu, L. Qi, A. Jolfaei, W. Ding, M.R. Khosravi, Intrusion detection for maritime transportation systems with batch federated aggregation, IEEE Trans. Intell. Transp. Syst. 24 (2) (2023) 2503–2514.
[3] I. Homoliak, K. Malinka, P. Hanacek, ASNM datasets: a collection of network attacks for testing of adversarial classifiers and intrusion detectors, IEEE Access 8 (2020) 112427–112453.
[4] G. Karatas, O. Demir, O. Sahingoz, Increasing the performance of machine learning-based idss on an imbalanced and up-to-date dataset, IEEE Access 8 (2020) 32150–32162.
[5] G. Apruzzese, M. Andreolini, M. Marchetti, A. Venturi, M. Colajanni, Deep reinforcement adversarial learning against botnet evasion attacks, IEEE Trans. Netw. Serv. Manag. 17 (4) (2020) 1975–1987.
[6] Z. Zhang, Y. Zhang, D. Guo, L. Yao, Z. Li, SecFedNIDS: robust defense for poisoning attack against federated learning-based network intrusion detection system, Future Gener. Comput. Syst. 134 (2022) 154–169.
[7] C. Pimsarn, T. Boongoen, N. Iam-On, N. Naik, L. Yang, Strengthening intrusion detection system for adversarial attacks: improved handling of imbalance classification problem, Complex Intell. Syst. 8 (2022) 4863–4880.
[8] J. Xu, X. Wang, Z. Cai, L. Yang, L. Jing, Informative instance detection for active learning on imbalanced data, in: Proceedings of IEEE International Joint Conference on Neural Networks, 2019, pp. 1–7.
[9] P. Panwong, T. Boongoen, N. Iam-On, Improving consensus clustering with noise-induced ensemble generation, Expert Syst. Appl. 146 (2020) 113–138.
[10] D. Karaboga, B. Gorkemli, C. Ozturk, N. Karaboga, A comprehensive survey: artificial bee colony (ABC) algorithm and applications, Artif. Intell. Rev. 42 (1) (2014) 21–57.
[11] M.A. Awadallah, M.A. Al-Betar, A.L. Bolaji, E.M. Alsukhni, H. Al-Zoubi, Natural selection methods for artificial bee colony with new versions of onlooker bee, Soft Comput. 23 (2019) 6455–6494.
[12] B. Kolukisa, B.K. Dedeturk, H. Hacilar, V.C. Gungor, An efficient network intrusion detection approach based on logistic regression model and parallel artificial bee colony algorithm, Comput. Stand. Interfaces 89 (2024) 103808.
[13] M. Mazini, B. Shirazi, I. Mahdavi, Anomaly network-based intrusion detection system using a reliable hybrid artificial bee colony and Adaboost algorithms, J. King Saud Univ, Comput. Inf. Sci. 31 (4) (2019) 541–553.
[14] P.I. Priyadarsini, ABC-BSRF: Artificial Bee Colony and Borderline-SMOTE RF algorithm for intrusion detection system on data imbalanced problem, in: Proceedings of International Conference on Computational Intelligence and Data Engineering, 2020, pp. 15–29.
[15] W.C. Lin, C.F. Tsai, Y.H. Hu, J.S. Jhang, Clustering-based undersampling in class-imbalanced data, Inf. Sci. 409–410 (2017) 17–26.
[16] C. Seiffert, T. Khoshgoftaar, J.V. Hulse, A. Napolitano, Rusboost: a hybrid approach to alleviating class imbalance, IEEE Trans. Syst. Man Cybern., Part A 40 (1) (2010) 185–197.
[17] M. Tahir, J. Kittler, F. Yan, Inverse random under sampling for class imbalance problem and its application to multi-label classification, Pattern Recognit. 45 (10) (2012) 3738–3750.
[18] G. Ahn, Y.J. Park, S. Hur, A membership probability-based undersampling algorithm for imbalanced data, J. Classif. 38 (2021) 2–15.
[19] A. Guzman-Ponce, J.S. Sanchez, R.M. Valdovinos, J.R. Marcial-Romero, DBIG-US: a two-stage under-sampling algorithm to face the class imbalance problem, Expert Syst. Appl. 168 (2021) 114301.
[20] W.C.L.C.F. Tsai, Y.H. Hu, G.T. Yao, Under-sampling class imbalanced datasets by combining clustering analysis and instance selection, Inf. Sci. 477 (2019) 47–54.
[21] V.P.K. Turlapati, M.R. Prusty, Outlier-SMOTE: a refined oversampling technique for improved detection of COVID-19, Intell.-Based Med. 3–4 (2020) 100023.
[22] R. Ahsan, W. Shi, J.P. Corriveau, Network intrusion detection using machine learning approaches: addressing data imbalance, IET Cyber-Phys. Syst., Theory Appl. 7 (2022) 30–39.
[23] B. Molina-Coronado, U. Mori, A. Mendiburu, J. Miguel-Alonso, Survey of network intrusion detection methods from the perspective of the knowledge discovery in databases process, IEEE Trans. Netw. Serv. Manag. 17 (4) (2020) 2451–2479.
[24] R. Abdulhammed, M. Faezipour, A. Abuzneid, A. Abumallouh, Deep and machine learning approaches for anomaly-based intrusion detection of imbalanced network traffic, IEEE Sens. Lett. 3 (1) (2019) 1–4.
[25] Y.J. Lee, Y.R. Yeh, Y.C.F. Wang, Anomaly detection via online oversampling principal component analysis, IEEE Trans. Knowl. Data Eng. 25 (7) (2013) 1460–1470.

[26] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, Variational data generative model for intrusion detection, Knowl. Inf. Syst. 60 (1) (2019) 569–590.
[27] X. Ma, W. Shi, AESMOTE: adversarial reinforcement learning with SMOTE for anomaly detection, IEEE Trans. Netw. Sci. Eng. 8 (2) (2021) 943–956.
[28] Akashdeep, I. Manzoor, N. Kumar, A feature reduced intrusion detection system using ANN classifier, Expert Syst. Appl. 88 (2017) 249–257.
[29] A. Chandra, S.K. Khatri, R. Simon, Filter-based attribute selection approach for intrusion detection using k-means clustering and sequential minimal optimization technique, in: Proceedings of Amity International Conference on Artificial Intelligence, 2019, pp. 740–745.
[30] N. Farnaaz, M.A. Jabbar, Random forest modeling for network intrusion detection system, Proc. Comput. Sci. 89 (2016) 213–217.
[31] N.V. Sharma, N.S. Yadav, An optimal intrusion detection system using recursive feature elimination and ensemble of classifiers, Microprocess. Microsyst. 85 (2021) 104293.
[32] A.K. Shrivas, A.K. Dewangan, An ensemble model for classification of attacks with feature selection based on KDD99 and NSL-KDD data set, Int. J. Comput. Appl. 99 (15) (2014) 8–13.
[33] M.R. Parsaei, S.M. Rostami, R. Javidan, A hybrid data mining approach for intrusion detection on imbalanced NSL-KDD dataset, Int. J. Adv. Comput. Sci. Appl. 7 (6) (2016) 20–25.
[34] A. Tesfahunand, D.L. Bhaskari, Intrusion detection using random forests classifier with SMOTE and feature reduction, in: Proceedings of International Conference on Cloud Ubiquitous Computing and Emerging Technology, 2013, pp. 127–132.
[35] V. Hajisalem, S. Babaie, A hybrid intrusion detection system based on ABC-AFS algorithm for misuse and anomaly detection, Comput. Netw. 136 (2018) 37–50.
[36] X. Gao, C. Shan, C. Hu, Z. Niu, Z. Liu, An adaptive ensemble machine learning model for intrusion detection, IEEE Access 7 (2019) 82512–82521.
[37] Y. Zhou, G. Cheng, S. Jiang, M. Dai, Building an efficient intrusion detection system based on feature selection and ensemble classifier, Comput. Netw. 174 (2020) 107247.
[38] P. Tatongjai, T. Boongoen, N. Iam-On, N. Naik, L. Yang, Classification of adversarial attacks using ensemble clustering approach, Comput. Mater. Continua 74 (2) (2022) 2479–2498.
[39] Z. Xu, H. Khan, R. Muresan, TMorph: a traffic morphing framework to test network defenses against adversarial attacks, in: Proceedings of International Conference on Information Networking, 2022, pp. 18–23.
[40] M. Barreno, B. Nelson, A. Joseph, J. Tygar, The security of machine learning, Mach. Learn. 81 (2) (2010) 121–148.
[41] L. Nanni, C. Fantozzi, N. Lazzarini, Coupling different methods for overcoming the class imbalance problem, Neurocomputing 158 (2015) 48–61.
[42] P. Keerin, N. Iam-On, J.J. Liu, T. Boongoen, Q. Shen, Summarising multiple clustering-centric estimates with OWA operators for improved KNN imputation on microarray data, Fuzzy Sets Syst. 473 (2023) 108718.
[43] T. Boongoen, N. Iam-On, J. Mullaney, Providing contexts for classification of transients in a wide-area sky survey: an application of noise-induced cluster ensemble, J. King Saud Univ, Comput. Inf. Sci. 34 (8) (2022) 5007–5019.
[44] K. He, D.D. Kim, M.R. Asghar, Adversarial machine learning for network intrusion detection systems: a comprehensive survey, IEEE Commun. Surv. Tutor. 25 (1) (2023) 538–566.
[45] I. Debicha, B. Cochez, T. Kenaza, T. Debatty, J.M. Dricot, W. Mees, Adv-Bot: realistic adversarial botnet attacks against network intrusion detection systems, Comput. Secur. 129 (2023) 103176.
[46] A. Venturi, G. Apruzzese, M. Andreolini, M. Colajanni, M. Marchetti, Drelab-deep reinforcement learning adversarial botnet: a benchmark dataset for adversarial attacks against botnet intrusion detection systems, Data Brief 34 (2021) 106631.
[47] M. Alazab, S.P. RM, P.K.R. Maddikunta, T.R. Gadekallu, Q.V. Pham, Federated learning for cybersecurity: concepts, challenges, and future directions, IEEE Trans. Ind. Inform. 18 (5) (2022) 3501–3509.
[48] J. Liu, S. Anavatti, M. Garratt, K.C. Tan, H. Abbass, A survey, taxonomy and progress evaluation of three decades of swarm optimization, Artif. Intell. Rev. 55 (5) (2022) 3607–3725.
[49] K. Sriwanna, T. Boongoen, N. Iam-On, Graph clustering-based discretization of splitting and merging methods (graphs and graphm), Hum.-Cent. Comput. Inf. Sci. 7 (1) (2017) 1–39.
[50] P. Keerin, W. Kurutach, T. Boongoen, A cluster-directed framework for neighbour based imputation of missing value in microarray data, Int. J. Data Min. Bioinform. 15 (2) (2016) 165–193.