Computers & Security 123 (2022) 102918

Contents lists available at ScienceDirect

Computers & Security

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

A GPU-based machine learning approach for detection of botnet attacks

Michal Motylinski[a], Áine MacDermott[a,*], Farkhund Iqbal[b], Babar Shah[b]

[a] School of Computer Science and Mathematics, Liverpool John Moores University, Liverpool, UK
[b] College of Technological Innovation, Zayed University, United Arab Emirates

ARTICLE INFO

Article history:
Received 18 February 2022
Revised 11 August 2022
Accepted 10 September 2022
Available online 14 September 2022

Keywords:
Internet of Things
Machine learning
Random forest
Feature selection
Attack detection
Classification

ABSTRACT

Rapid development and adaptation of the Internet of Things (IoT) has created new problems for securing these interconnected devices and networks. There are hundreds of thousands of IoT devices with underlying security vulnerabilities, such as insufficient device authentication/authorisation making them vulnerable to malware infection. IoT botnets are designed to grow and compete with one another over unsecure devices and networks. Once infected, the device will monitor a Command-and-Control (C&C) server indicating the target of an attack via Distributed Denial of Service (DDoS) attack. These security issues, coupled with the continued growth of IoT, presents a much larger attack surface for attackers to exploit in their attempts to disrupt or gain unauthorized access to networks, systems, and data. Large datasets available online provide good benchmarks for the development of accurate solutions for botnet detection, however model training is often a time-consuming process. Interestingly, significant advancement of GPU technology allows shortening the time required to train such large and complex models. This paper presents a methodology for the pre-processing of the IoT-Bot dataset and classification of various attack types included. We include descriptions of pre-processing actions conducted to prepare data for training and a comparison of results achieved with GPU accelerated versions of Random Forest, k-Nearest Neighbour, Support Vector Machine (SVM) and Logistic Regression classifiers from the cuML library. Using our methodology, the best-trained models achieved at least 0.99 scores for accuracy, precision, recall and f1-score. Moreover, the application of feature selection and training models on GPU significantly reduced the training and estimation times.

1. Introduction

The Internet of Things (IoT) represents the seamless merging of the real and digital world, with new devices being created that store and pass around data. New frameworks, many interconnected devices, and a plethora of applications (allowing communication with said devices) make it difficult to develop and maintain robust security solutions. The growing numbers of IoT devices make them a very attractive target for threat actors who aim to use them to access other devices and a form a larger network. According to Kaspersky's Threat Report, "*the IoT will become one of the main targets of cyber-attacks in the near future*" (Kaspersky, 2022). Malicious software, or malware, arguably constitutes one of the most significant categories of threats to computer systems. With nearly 12,000 new instances of malware being created everyday detection of such threats is one of the most essential problems that require a solution (G Data, 2022). With the number of malware families targeting these IoT devices and systems is ever increasing, IoT botnets are designed to grow and compete with one another over unsecure devices. An IoT Botnet is also a collection of various IoT devices such as routers, wearables and embedded technologies infected with malware. Much of a botnet's power comes from the number of devices that make it up. As such, this malware allows an attacker to control all the connected devices. There are three distinct architectures that characterize most botnets. In the centralized network all bots connect to the Command-and-Control server (C&C). The main characteristic of this type is that automated commands are sent from C&C to the bots via IRC or HTTP channels. Direct communication means low latency of such architecture but also dependency on the C&C which if discovered will provide information about all botnets in the network. The second type of architecture is a decentralized model which does not have a central

[*] Corresponding author.
E-mail address: a.m.macdermott@ljmu.ac.uk (Á. MacDermott).

https://doi.org/10.1016/j.cose.2022.102918

point of failure. In this setup each bot is both client and server and use peer to peer (P2P) communication protocols as a means of connecting with other machines. In the hybrid approach the compromise between centralized and decentralized approach allows to keep relatively low latency and keep botnets secure from detection by using P2P protocols for communication (Miller et al., 2016). Early IoT malware families like Gafgyt and the original Mirai family leveraged default or weak passwords to attack devices. Whereas current versions of botnet have new functionalities, and propagation methods utilise Tor proxy functions to provide the IP servers' address. Botnets are mainly propagated through weak Telnet passwords – a common issue on IoT devices – and through exploiting three vulnerabilities. The Gafgyt botnet actively targets vulnerable D-Link and IoT devices including remote code execution flaws (CVE-2019–16,920) in D-Link devices; a remote code execution vulnerability in Liferay enterprise portal software (for which no CVE is available); and a flaw (CVE-2019-19,781) in Citrix Application Delivery Controllers (Threatpost, 2021).

The best strategy against IoT botnets is to secure against their threat, detect their presence in a timely manner, and ultimately limit their resources (by reducing the number of unsecure devices from which they could derive their power). Intrusion Detection Systems (IDS) are used to monitor network traffic and detection sign of intrusion. The detection may be according to the signatures of executable malwares or according to the signatures of malicious network traffic generated by malware. Signature-based approaches detect malicious packets by looking at specific patterns and signatures of the given threat. A major problem with this approach is that it requires frequent updates of the intruder's database and is unable to detect unknown attacks. Anomaly-based detection focuses on learning trustworthy signatures (and behaviours) and uses this knowledge to pass only legitimate traffic. If an IDS detects an unusual pattern in analysed traffic, then the particular packets will be flagged. However, the main problem with this approach is that new legitimate traffic can also be flagged because the algorithm had not learned it yet, with an increasing amount of false positive alerts. Any action, like sweeping or probing, creates a signal in the network anomaly-based IDS which can detect such actions.

Machine learning has become a vital technology for cybersecurity and threat detection (Xin et al., 2018; Azwar et al., 2018). Machine learning for intrusion detection can solve many challenges such as speed and computational time and develop accurate IDS. While the application of machine learning for classification or detection of attacks has been covered in many academic works, we have not yet seen an attempt to implement acceleration technologies to boost the performance of the models and essentially create a more viable solution for environments where frequent retraining of the algorithm is necessary. There are various frameworks available for an acceleration of the machine learning models. In this paper we will focus on the implementation of RAPIDS libraries such as cuDF and cuML. The aforementioned libraries allow the use of GPU for machine learning tasks which may provide increased performance due to significantly greater bandwidth and better computation capabilities of GPU over CPU (Medium, 2021). Due to the difference in architecture between CPU (typically 4–8 cores) and GPU (hundreds of smaller cores) parallelization of tasks can be applied when working on the latter. Using the CUDA platform for parallel programming, the general computing tasks can be drastically sped up by breaking down one big task into hundreds of little chunks.

Our research is focused on increasing the speed of the detection while sustaining an acceptable level of detection. Our methodology involves pre-processing, feature selection and application of GPU-accelerated machine learning models which results in an improvement over currently used methods. These methods are explained within our methodology section and comparison to related works is conducted within the Results section. Our approach differs from other works in the field as we decided to create new features from the existing dataset. Moreover, in contrast to other works, we decided to test fast computing algorithms and their impact on accuracy, training, and prediction time of the models.

The novel contributions of our work are as follows:

- Application of GPU-based accelerated machine learning models,
- Generation of new features and application of permutation importance method for feature selection and interpretability of models,
- Improvement of both training and prediction times in comparison to other works in the field,
- Retaining high accuracy and robustness of the models similar to previous academic works.

The paper is organised as follows: In Section 2 we provide background on attacks against IoT devices and related works utilising machine learning. In Section 3 we detail our methodology, and present results and discussion of our findings Section 4. Future work and concluding remarks are presented in Section 5.

2. Related work

Machine learning algorithms use historical data as an input to predict new output values. Machine learning can monitor systems and respond to changes in the behaviour, protecting against threats through pattern detection, real-time threat monitoring, vulnerability mapping and penetration testing. Machine learning methods have seen increased use in the last decades due to the rapid development of various technologies and the growing computing capabilities of computers. The introduction of GPU for machine learning has introduced new possibilities allowing researchers to solve issues that previous hardware could not handle due to expensive operations or significant time-consuming processes of model training. Machine learning models are known for great prediction capabilities and are used for a variety of classification, pattern recognition and detection tasks.

The difference between cyber security and other fields is that the attackers and threat actors do not behave in a predictable or statistically consistent way. The goal of an attacker is to remain hidden and so all their activities are evasive. As such, an attack performed by one attacker may look completely different to the same attack performed by a different attacker. This means that many machine learning models cannot be widely used and that the models and algorithms must be adapted to different conditions and behavioural parameters. IDS play a crucial role in defending networks by monitoring traffic for malicious activities.

The majority of the solutions tackling traffic detection problems focus solely on the accuracy, however, training and prediction time is also important. Within this section we explore the solutions proposed for the classification of attacks using the IoT-Bot dataset UNSW Canberra (2022) and their parameters for detection.

2.1. Current solutions

Koroniotis et al. (2019) used the machine learning classifier Support Vector Machine (SVM) and two deep learning predictors Recurring Neural Network (RNN) and Long Short-Term Memory (LSTM). Their experiments were conducted on a 5% sample of data which contained around 3 million records. The authors derived new features from the existing data. Using a correlation coefficient, the researchers extracted the 10 best features that were used to train a model and compared against training on a full set of features. The SVM trained on all features achieved the best results with accuracy of 99 and 100% recall, however all predictors had a very similar performance. The training time of SVM was 110 min. The results of this research show that while SVM and neural networks have extremely high accuracy they are also very slow to train and require a significant amount of data.

Oreški et al. (2020) used a different approach to the selection of best features called 'Search and Testing for Understandable Consistent Contrast' (STUCCO). With this approach, the authors were able to select different features compared to the work of Koroniotis et al. (2019). The authors implemented the SVM model to train on the input data. The model has achieved >0.99 scores for accuracy, precision, recall and f1 score.

In Shafiq et al. (2020a), the authors proposed a novel method for feature selection called 'CorrAUC' and applied it to the Bot-IoT dataset. The new technique selected a set of five features that described the dataset well enough to be used for training. The approach trained Decision Tree, SVM, Naive Bayes and Random Forest classifiers and compared their performance on a created test set. With the exception of Naive Bayes, all classifiers achieved high accuracy, specificity, sensitivity, and precision scores in most cases. The results indicate that Random Forest performance was slightly better, and the accuracy was above 99% which is similar to the previous research, however, recall scores for data theft and keylogging theft were 0.50 and 0.89 accordingly.

In their work, Javed et al. (2020) proposed the use of an AdaBoost classifier for the detection of botnet attacks. The authors used a publicly available "takata" dataset for their research. The applied method for feature extraction allowed for deriving a set of 10 highly correlated features out of the initial list of 55 features. For comparison, the authors applied a decision tree, probabilistic neural network, and sequential minimal optimization algorithms. The evaluation results indicate that AdaBoost has the highest accuracy and robustness out of all four architectures tested. The proposed approach involves feature selection using the information gain method and then the implementation of the AdaBoost classifier.

Churcher et al. (2021) performed a comprehensive analysis of attack classification using many common algorithms from the scikit-learn library like KNN, SVM, Random Forest or Naïve Bayes. The researchers conducted 2 types of experiments a binary classification of malicious traffic and a multiclass classification of various attacks. Various weights were applied to the classifiers to change the bias towards classes. Random forest was the best performing algorithm for binary classification tasks while KNN and ANN models had better performance classifying various attack types. Random Forest had perfect metric scores in a binary task and 0.95 scores in a multiclass task.

Shafiq et al. (2020b) tested various machine learning models in search of the most effective solution for IoT botnet detection. The IoT Botnet dataset from Koroniotis et al. (2019) has been used to conduct this research. The authors selected Naïve Bayes, BayesNet, Decision Tree, Random Forest and Random Tree and applied the Bijective Soft Set technique to choose the best classifier. The results of this research show that all algorithms have a high accuracy and recall rate of >0.98. Taking into consideration the time required to train the algorithms in this research Native Bayes had the best performance.

In Alsamiri and Alsubhi (2019), the authors used IoT-Botnet pcap files to generate a new set of features by using the CICFlowMeter tool to extract flow-based features. The authors selected 13 generated features for the training of various models including Random Forest, k-Nearest Neighbour and Naive Bayes. The results presented in the work indicate that Random Forest has the best performance for most of the attack types (95%–100%) with KNN having slightly lower accuracy. In their work (Garre et al., 2021) proposed a novel approach for the detection of SSH botnet infections. The authors generated their own dataset capturing information from various honeypots deployed across the world. For traffic classification, four algorithms were used namely: Decision Tree, Random Forest, SVM and Native Bayes. Experimental results showed that Random Forest had better performance achieving 95.7% accuracy and 93.9% recall scores.

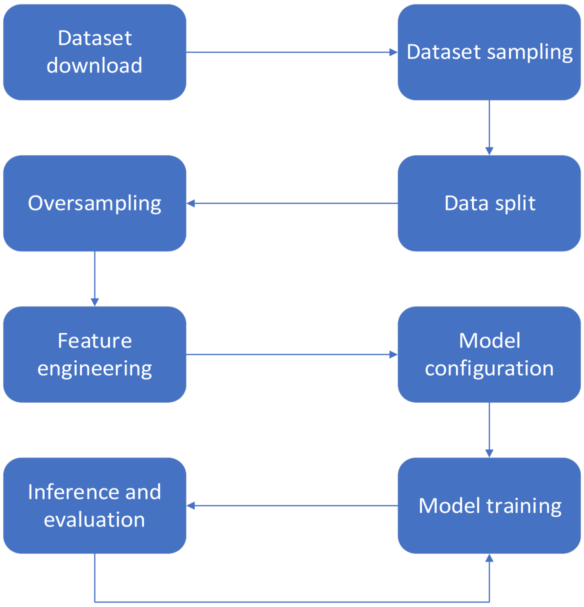

Fig. 1. IDS machine learning model development pipeline.

2.2. Summary of related work

Our analysis of related works indicates that SVM is one of the most commonly used classifiers for its great accuracy in comparison to many other methods. Ensemble learning algorithms, however, tend to perform better than SVM, especially Random Forest (RF) which became a state of the art in many domains in recent years (Vakili et al., 2020; Sujatha and Mahalakshmi, 2020). The tests conducted on various datasets conclude that RF is not only more accurate than SVM in most cases but also requires significantly less training time and provides faster prediction. The speed of the algorithm training and prediction is important for their industry use because the less time and resources it is necessary to develop a good model the sooner it can be deployed. It is extremely important as machine learning models for IoT detection must be regularly updated to keep up with new threats which means frequent model retraining.

3. Proposed methodology

In this section, we provide our methodology used to develop a model capable of discriminating different types of attacks on IoT devices. The model development pipeline (presented in Fig. 1) begins from the data processing stage which involved acquisition of the dataset from a public repository and sampling a smaller set containing enough information to train the machine learning models. As part of pre-processing, the data is split into appropriate sets for model training and evaluation. Next, an oversampling ratio is applied to parts of the data. One of the most important aspects of our approach is the creation of a new set of features that are derived in a feature engineering process. Then all of the features undergo a selection process which results in much smaller set of best features that are used to train the models. Following this stage is the hyperparameter configuration of the model. This involves the

Table 1
Dataset attack class distribution.

Attack category	Subcategory	Number of entries
DDoS	HTTP	19,738
	TCP	19,547,104
	UDP	18,964,396
DoS	HTTP	29,680
	TCP	12,315,619
	UDP	20,658,630
Reconnaissance	OS Fingerprint	1,433,189
	Service Scan	356,285
Theft	Data Exfiltration	114
	Keylogging	1464

retraining and evaluation process which occurs until satisfying results are achieved. Our approach involves implementation of GPU accelerated algorithms that allow significantly faster model training and prediction.

The remainder of this section will include methodology details and experiment design via description of the dataset used for our experiments, justification of feature selection, our approach to uneven class distribution, choice of classifiers and selection of metrics used for evaluation of the models.

3.1. Dataset

We have chosen the most recent iteration of IDS datasets from The University of New South Wales Canberra (UNSW) at the Australian Defence Force Academy – ‘*Bot-IoT*’- UNSW Canberra (2022). The data was created in a Cyber Range Lab in a realistic environment (Koroniotis et al., 2017, 2019; Koroniotis and Moustafa, 2020). From PCAP files, a set of features were extracted and saved in various formats. We are using CSV files, with the overall size at 16.7 GB (there are 72 million records in the dataset). 9063 of the entries represent normal traffic. This data is used for a binary classification of malicious and non-malicious traffic. Each entry is described as belonging to one of the main attack categories and further split into a subcategory. Table 1 presents the distribution of attacks according to category and subcategory.

Dataset sample

While the dataset authors (UNSW Canberra, 2022) (Koroniotis et al., 2017) provided a pre-processed subset of data with nearly three million entries, the distribution of attack types is very unbalanced. We decided to create our own subset which consists of a more equal representation of all attack types. We concluded that 100,000 occurrences per attack would be sufficient to train an accurate solution. A fixed-size sample of random values was taken for every class if a number of occurrences exceeded the limit.

Experiment environment

In this experiment no physical setup is made to create a malicious traffic. Instead, a “*Bot-IoT*” *dataset* - well known benchmark - is used to train and test the algorithms. *The* training of all models was performed on AMD Ryzen 7 2700X Eight-Core Processor (4.15 GHz). It is important to note that every CPU based training used all processors for training which significantly improved the training time of the models. A single NVIDIA GeForce RTX 2080 graphics card with 8GB of VRAM was used for the training of accelerated variants of machine learning algorithms.

Feature selection

To reduce training and prediction times, we removed features that had little or no impact on the prediction capabilities of the algorithms. This is achieved by implementing the permutation importance technique. The permutation feature importance provides feedback about which feature in a dataset had the least importance. This is done by randomly shuffling feature value which causes a decrease in model score. The procedure breaks relationship between the feature and the target which shows how dependant the model is on the particular feature. The conducted tests showed that the best features obtained with this method work well with all estimators used in this study. Moreover, using the original set of features we have calculated new features to increase the robustness of the model. The results of this process are shown in the following subsections where we remark on class distribution, estimators, and evaluation metrics.

Class distribution

Uneven distribution of classes is a very common issue in machine learning. In fact, it is very difficult to find perfectly even datasets especially with thousands or millions of records. Depending on the scale of irregularity this can be a serious problem, and in some cases lead to very poor results of prediction. While some classifiers like decision trees, logistic regression and SVM can work with imbalanced data, they will most likely fail when there is a high disproportion of classes. In order to tackle the problem of imbalanced attributes two methods can be employed: oversampling and under-sampling. Application of the former technique requires instances of the under-represented data to be copied. Under-sampling on the other hand can be applied by deleting instances of the major class.

It is generally advised to use oversampling on small datasets and under-sampling when there is a lot of data so removal of values will not have a negative impact on the model. As the produced dataset sample was still imbalanced, we have decided to apply oversampling to the minor class to eliminate the bias. We have chosen one of the most widely used methods called the Synthetic Minority Oversampling Technique (SMOTE).

Algorithms used

Based on previous academic research in the field and our own experience we have decided to use Random Forest (RF), SVM, Logistic Regression (LR) and k-Nearest Neighbour estimators. For our experiments, we have chosen GPU accelerated versions of the classifiers from the cuML library. While Scikit-learn implementations are considered state-of-the-art and are used in most research works on IoT-bot detection they can only utilize CPU which training, and prediction times are a major drawback. The GPU accelerated algorithms are still in development, thus many of the features included in the documentation are not yet supported. RAPIDS algorithms tend to perform worse than their scikit-learn counterparts on default settings, thus hyperparameter optimization was necessary to obtain satisfactory results (cuML, 2022).

RF is an ensemble learner that implements multiple weak learners (decision trees) using specific rules and then integrates results from all of them generating the final prediction. Each tree is trained on a random subset of features which breaks the correlation between them improving the prediction capability of the model. RF is considered as a state-of-the-art algorithm for its prediction accuracy tested on many different datasets as well as the very short time necessary to train the model (Breiman, 2001; Zhang et al., 2017; Nanni et al., 2015). k-Nearest Neighbour (KNN) is a supervised machine learning algorithm that assumes that similar elements exist in close proximity. KNN can be used for both classification and regression problems. Classification is performed by looking at the closest neighbour to the chosen K value of the same class. As the name suggests the most important hyperparameter in KNN is the number of neighbours (n_neighbors). Other parameters such as distance metrics and weights of neighbours

Table 2
Hyperparameter values for each algorithm.

Algorithm	Parameter	Default	Tuned
Random Forrest (RF)	max_depth	16	18
	n_bins	8	17
k-Nearest Neighbour (KNN)	N_neighbours	1	3
Support Vector Machines (SVM)	C	1	60
Logistic Regression (LR)	penalty	l2	l1
	tol (tolerance)	1e-4	1e-5

can also change the prediction significantly depending on the task and data composition. The KNN is a fast algorithm to train, however, its major drawback is significantly slower estimation time (Altman, 1991).

The SVM is also a very popular model which is often used to solve many classification problems. The most important parameter in SVM is the kernel which controls how the input variables are projected. SVM divides n_dimensional space into two distinct regions for output classes. The algorithm is trying to find a hyperplane during training that best separates the output classes. In the case of binary problem hyperplane is a single line. SVM algorithm is commonly used for its high prediction rate, however, a major drawback of this method is training time which is significantly higher than RF or KNN (Abdiansah and Wardoyo, 2015).

Logistic regression (LR) is one of the most common models used for binary classification. LR is rarely used for intrusion detection tasks, however, its performance for binary problems is usually on par with other state-of-the-art algorithms. LR hyperparameters can provide some improvement to the performance of the model. The regularisation (penalty) and C parameter usually have the greatest impact on the model performance (Pohar et al., 2004).

Initial training iterations were conducted using default parameters; however, the results were poor, thus hyperparameter tuning was applied to all four models. For the RF model 2 parameters were tuned: max depth and number of bins. The former represents the depth of every tree which determines the number of splits. Generally, more splits allow the model to capture more information, however the convergence time increases. The cuML RF implements a histogram-based method for split determination. The size of histograms can be tuned using number of bins parameter. This is especially useful for larger problems with highly skewed input data. The only hyperparameter tuned for KNN algorithm was the number of neighbours or K number which indicates the count of the nearest neighbours. In the case of SVM classifier tuning of C parameter provided the best results. The C parameter is a penalty that determines the influence of the misclassification on the decision function. The higher the penalty enforces a smaller error margin for decision function choosing hyperplane while lower value encourages a larger error margin for the cost of model's accuracy. Two parameters were tuned for the LR namely penalty and tolerance. The penalty type refers to the regularisation method that reduces parameters and simplifies the model to avoid overfitting. The tolerance value determines when to stop the training. Depending on the task and input data larger values may cause algorithm to not converge. Table 2 presents the exact values of parameters chosen for each algorithm.

3.2. Evaluation metrics

To quantify the performance of the trained models the predicted values are assessed using evaluation metrics. Various metrics make different assumptions about the problem; thus, it is important to validate the outcome using multiple metrics. In this case we have decided to apply standard set of evaluation metrics to each estimator: accuracy, precision, recall and F1-score. Values for each metric are calculated from the confusion matrix of predictions. The accuracy is the ratio of the number of correct predictions to the total number of samples. The formula for accuracy is presented in (1). A True Positive (TP) is an outcome where the model correctly predicts the positive class. Similarly, a True Negative (TN) is an outcome where the model correctly predicts the negative class. A False Positive (FP) is an outcome where the model incorrectly predicts the positive class. A False Negative (FN) is an outcome where the model incorrectly predicts the negative class. Accuracy works best when the number of samples belonging to each class is equal, thus under-sampling should positively impact the score. These metrics will be used when analysing the performance of our improved approach and comparing to related works.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Precision (2) is the number of ground TP results divided by number of predicted positive results.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

Recall (3) is the number of correct positive results divided by the number of all positive samples from the class.

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

F1-score (4) is a mean between precision and recall that ranges between 0 and 1. F1-score indicates how robust the model is.

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{4}$$

Moreover, to test the speed of algorithms running on GPU the results are compared to the CPU counterparts from the scikit-learn library. The speed was measured in seconds and compiled results include mean speeds calculated from 10 training/test iterations per model.

4. Results and discussion

In this section, we present and discuss the results obtained in the conducted experiments. First, we cover the outcomes of a data processing pipeline developed for this project. Second, we discuss the results of binary detection of malicious traffic. Next, accuracy across all of the classes is presented. Finally, we discuss the impact of our project in comparison to other works covering the IoT botnet detection process.

4.1. Data pre-processing

During the pre-processing stage, we have created a small subset of data that provides enough information to the algorithms and shortens the training and prediction time. Because our research is focused on the binary classification of the traffic, we have decided that data will be derived according to the number of occurrences per attack type. Unlike the method used by the authors of a dataset, we have saved all occurrences of minority classes (below 100,000) to ensure a good representation of all attacks. As shown in Fig. 2, the Keylogging, Data Exfiltration, DoS HTTP and DDoS HTTP classes are underrepresented in the dataset. The significant difference in the number of samples may introduce bias towards majority classes reducing accuracy of the algorithms. To tackle this issue, we have decided to adjust the class distribution by oversampling the minority classes.

During our research we have applied various data splits to test their impact on the trained models. We have observed that an 80:20 split of the data provided the best results. After the split, an oversampling was performed on the training set. As a result,

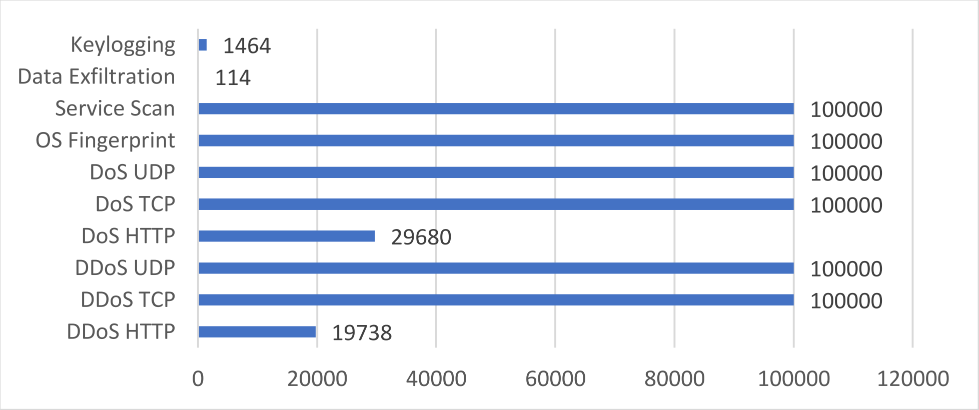

Fig. 2. Subset of attacks derived from the original dataset.

Table 3
Training sets utilised during research.

Attack	Normal	Oversampled
DDoS HTTP	15,790	80,000
DDoS TCP	80,000	80,000
DDoS UDP	80,000	80,000
DoS HTTP	23,744	80,000
DoS TCP	80,000	80,000
DoS UDP	80,000	80,000
Reconnaissance OS	80,000	80,000
Reconnaissance Service Scan	80,000	80,000
Theft Data Exfiltration	91	80,000
Theft Keylogging	1171	80,000
Normal traffic	7250	80,000

Table 4
Training input features.

Feature	Description
Pkts	Total count of packets in transaction
Bytes	Total number of bytes in transaction
State	Transaction state
Dur	Record total duration
Spkts	Source-to-destination packet count
Sbytes	Source-to-destination byte count
Sum	Total duration of aggregated records
Mean	Average duration of aggregated records
Pkts/Bytes	Packets to bytes ratio

Table 5
Binary detection of malicious traffic.

Algorithm	Accuracy	Precision	Recall	F1-score
RF	0.9995	0.99668	0.98479	0.99066
RF+SMOTE	0.99988	0.99857	0.99722	0.9979
KNN	0.99978	0.99715	0.99472	0.99593
KNN+SMOTE	0.98976	0.78658	0.99372	0.86145
SVM	0.99742	0.98688	0.91629	0.94875
SVM+SMOTE	0.99809	0.94285	0.99359	0.96681
LR	0.98874	0.82808	0.68046	0.73217
LR+SMOTE	0.95136	0.60798	0.96038	0.66429

additional records were added to the minority classes DDoS HTTP, DoS HTTP, and both ‘Theft attack’ types. This method evens out the class balance to 80,000 records per class. Table 3 presents the distribution of samples across different classes used for training the model.

Initial testing showed that the majority of the features do not impact the prediction capability of the models. As was mentioned in the methodology section, we have applied a feature importance algorithm to select a group of best features. Feature importance is defined as a decrease in model score when a feature is shuffled. The process of shuffling breaks the relationship between the target value and a feature, thus the drop in model score indicates how dependant the model is on the particular feature.

Fig. 3 presents the eight best original features selected for the training of all models. The features that had little to no impact on the model were removed from the input set and the remaining features were used to derive the new data.

To further increase the robustness of the models we have derived several features from the original values. Analysis showed that only the rate of packets to bytes had a meaningful impact on the algorithms, thus it was selected as one of the input values. SVM has benefited the most from the addition of a newly derived feature having its recall and f1-score increased by 0.1 scores. Table 4 presents the full set of features used for the training of all algorithms. Each feature and a description of the associated attribute is conveyed.

4.2. Binary malicious traffic detection

The first stage of our experiments involved the detection of malicious traffic. For this purpose, all attack entries were combined under the malicious traffic label, while normal traffic remained as a second class. The results of this classification are shown in Table 5. The performance metrics show that all models achieve a high level of accuracy. Moreover, implementation of oversampling visibly improves the results. This is especially visible in the case of the SVM classifier. Analysis of the evaluation metrics shows that RF and KNN performed significantly better achieving near 100% accuracy and sensitivity. This means that the results of these two models are significantly more robust. While the SVM also had high accuracy, it is lacking precision and recall (which indicates false classification and reduces the overall robustness of the model). The worst-performing algorithm was LR which evaluation shows a significant number of false positive predictions even after parameter tuning.

It is also important to note that while both RF and KNN have very similar metric scores there are significant differences in the number of misclassified samples for both classes. Fig. 4 shows the confusion matrix of RF and KNN classifiers trained on an uneven data sample. The KNN performance is better, and the difference can be observed in a number of normal traffic misclassified samples. The KNN is clearly more sensitive and as a result smaller portion of the traffic is being misclassified.

Interestingly, training on the oversampled set generated very different results (shown in Fig. 5). Random Forest performance has increased, especially the classification capability of the benign traf-

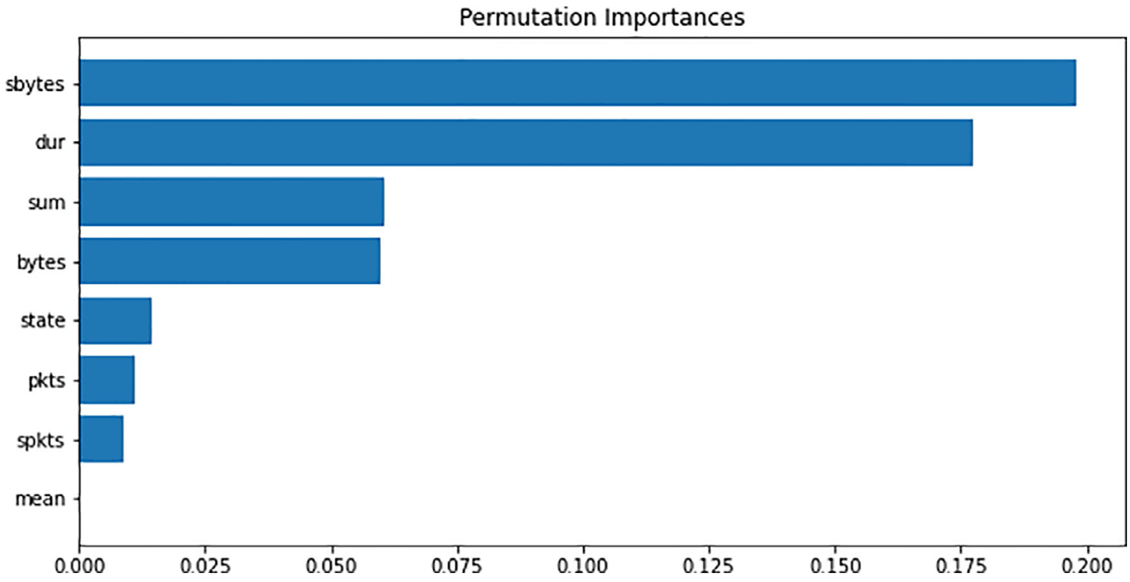

Fig. 3. Importance of features used for model training.

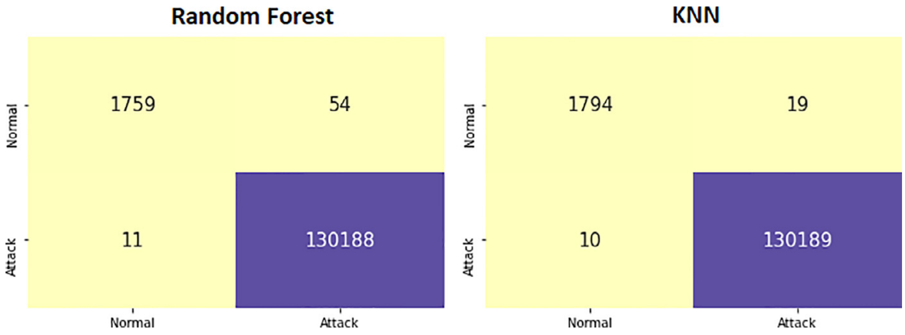

Fig. 4. Classification results for unbalanced set training.

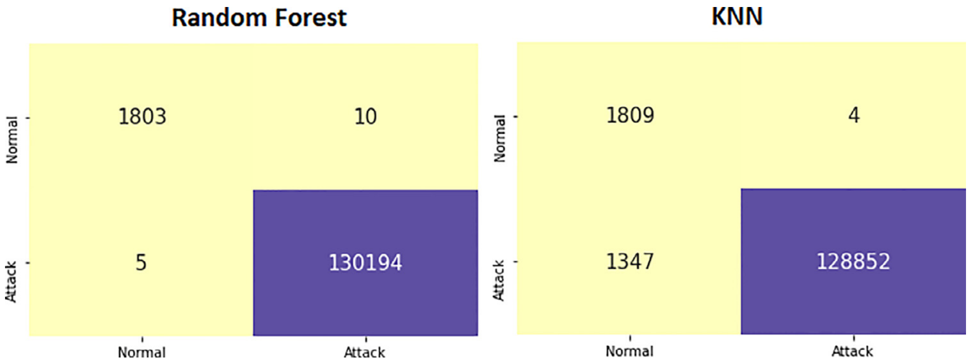

Fig. 5. Classification results for oversampled set training.

fic which is now nearly 100% accurate. Fig. 5 shows that oversampling the training data possibly introduced some bias impacting the prediction capability of KNN. The performance of KNN has decreased by a significant margin when we consider that the number of malicious traffic classified as benign has increased from 10 (as shown in Fig. 4) to 1347 (Fig. 5).

SVM tends to classify most of the traffic as malicious. This problem can be solved by adding additional features to the training set. Note however, the purpose of this research was to test the prediction capability on the smallest possible number of features, thus allowing fast training and estimation. The LR model has the highest number of misclassified traffic samples rendering it not a viable solution for an IDS.

The training time of KNN is significantly shorter than any other algorithm, however, the prediction time is much slower. This is because KNN does not generalize data in advance. While LR requires the least amount of time to make a prediction its accuracy and robustness is way too low to consider it a good option. SVM training time is significantly longer than any other algorithm which does not make it a viable solution for IDS which must be frequently retrained to include new threats. RF, while not the best in time metrics, is clearly the best algorithm as it grants the best prediction capability within reasonable training and prediction times.

We have compared both the training and prediction time of the algorithms running on GPU and CPU. As evident, the training times vary between different algorithms. The training of SVM is slow due

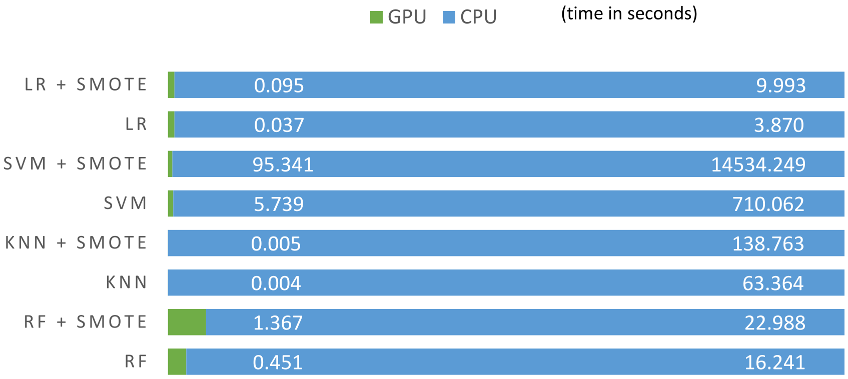

Fig. 6. Training time comparison on GPU and CPU.

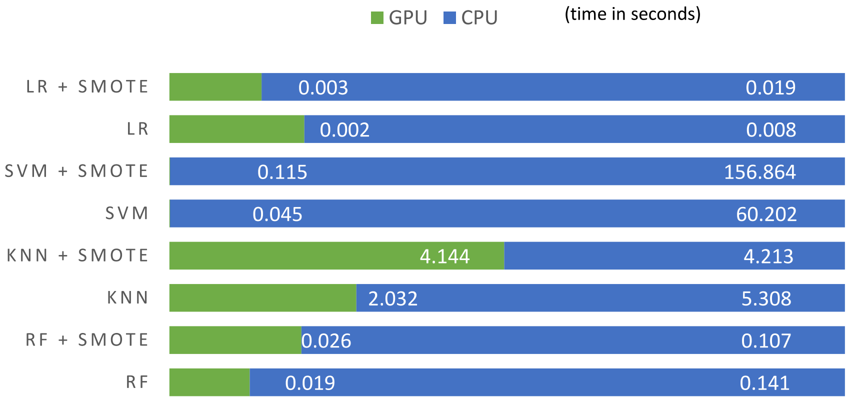

Fig. 7. The comparison of prediction time on GPU and CPU.

to the significant number of samples chosen and a non-linear kernel used. Future tests may involve smaller input sets to test the performance changes. The remaining algorithms converge below two minutes on CPU and in less than five seconds with GPU acceleration. Fig. 6 presents the results of a training time comparison which clearly show how much quicker the process of retraining can be when utilizing GPU.

In Fig. 7, we can see the estimation times for each algorithm trained. As we can observe, the KNN required a significantly longer estimation time because for every prediction it needs to scan all nearest neighbours in the completed training set. Again, a smaller training set would improve the prediction time however this might negatively impact the accuracy of the model. Another aspect worth noting is a significant improvement in estimation time when predicting with SVM classifier utilizing GPU. Overall, the GPU accelerated models are significantly faster in both training and prediction which in some cases may be a crucial factor.

4.3. Attack type detection

The second task of our study was to perform binary classification of every attack separately. Table 6 presents the results of binary detection of the attacks. RF is clearly the most accurate algorithm, however, KNN and SVM achieve similar results in most cases. LR was again the worst performing algorithm even with different parameter settings. The most important findings show that it was possible to achieve very high accuracy and robustness of the Random Forest classifier for all attack types. Implementation of oversampling reduced bias towards majority classes and as a result improved estimation of the models on the previously underrepresented attack types.

Table 6
Binary detection of individual attacks (F1-score only).

Attack type	RF	KNN	SVM	LR
DDoS HTTP	0.99939	0.99818	0.97920	0.91840
DDoS TCP	1	0.99954	0.99985	0.99593
DDoS UDP	0.99652	0.99894	0.99697	0.98567
DoS HTTP	0.99982	0.99803	0.95575	0.87587
DoS TCP	1	0.99939	1	0.99954
DoS UDP	0.99894	0.99864	0.99864	0.98582
OS Fingerprint	0.99939	0.99804	0.99474	0.85461
Service Scan	0.99729	0.99758	0.99562	0.97313
Data Exfiltration	1	0.96112	0.93174	0.63416
Keylogging	0.99901	0.99604	0.99506	0.97142

The application of algorithms on different benchmarks may provide interesting results and allow further improvements. Generation of new features can also be the answer for better performance and reduction of bins used in GPU accelerated RF which significantly increased training and prediction time. In future work we plan to apply other models from the cuML library to test their performance and compare them to the CPU-based versions.

4.4. Comparison with other works

In comparison to other academic works in the field, our method not only reduced training but it also significantly reduced prediction time by utilizing GPU. Specifically, dimensionality reduction provided a further improvement to the speed of the training and the evaluation process. The choice of a custom set retained more samples of the minority classes reducing bias and in turn provided more data for the models. As a result, it was possible to retain high performance as was shown in the results section.

Table 7
Comparison of algorithms performance.

Metric	Accuracy	Recall	Time (s)
Best RF - GPU	0.99988	0.99722	0.45
Best RF - CPU	0.99985	0.99666	16.24
Best SVM - GPU	0.99742	0.91629	5.74
Best SVM - CPU	0.99516	0.82839	710.06
Koroniotis et al. (2019) SVM	0.88373	0.88371	1270
Koroniotis et al. (2019) SVM all features	0.99988	1	6636.98
Koroniotis et al. (2019) RNN	0.99740	0.99749	8035
Koroniotis et al. (2019) RNN all features	0.97906	0.97908	6888.08
Koroniotis et al. (2019) LSTM	0.99741	0.97908	10,482.19
Koroniotis et al. (2019) LSTM all features	0.98057	0.98058	14,073.63
Shafiq et al. (2020b) RF	0.9999	1.00.	n/a
Alsamiri and Alsubhi (2019) RF	0.98	0.98	27.0328

We compared our improved GPU-based machine learning approach for detection of botnet attacks with related works of Koroniotis et al. (2019), (Shafiq et al., 2020b), and Alsamiri and Alsubhi (2019). Specifically, we analysed and compared our algorithms in terms of accuracy, recall and time. The accuracy comparison looks at the portion of correctly classified samples, whereas recall is to do with the correctly identified positive classes from the actual malicious traffic. In Table 7 the comparison of malicious traffic detection with the results obtained by models of authors Koroniotis et al. (2019), (Shafiq et al., 2020b), and Alsamiri and Alsubhi (2019) is presented. The former trained SVM classifier, RNN and LSTM networks use 5% of the original data and ten selected features. The authors of the second work implemented algorithms for selection of best features and tested the results by training various models. The best performance was achieved using RF; thus, the results are included in the comparison. Alsamiri and Alsubhi (2019) generated eighty new features from the original pcap files and selected seven for model training. The RF algorithm accomplished the best results having high accuracy and recall.

The most significant improvement of our solution can be seen in the training time which application of accelerated machine learning algorithms decreased considerably. The accuracy and robustness of our best algorithm are comparable to other authors results. In terms of accuracy developed models are outperformed slightly by Koroniotis et al. (2019) and their RNN architecture and (Shafiq et al., 2020b) with their RF model. As can be observed, the training time of the GPU-based models is significantly shorter outperforming all other architectures by a large margin.

The tested model's performance is on par with other works results with significant time improvement. Faster training allows for more frequent retraining of the model and updates of the system. This is especially important in production where quick model deployment allows to save resources and well optimised training pipelines are essential. The accelerated versions of machine learning algorithms also provide faster prediction which can be crucial in the fast identification of a threat.

5. Conclusions and future work

This paper presents our research into the application of GPU-based accelerated machine learning models. Four types of machine learning algorithms were compared in terms of accuracy, precision, recall, F1-score as well as computation time required to train the model and perform prediction. The experimental results show that the proposed data pre-processing and feature selection methods improve the training and prediction durations while maintaining the high performance of the estimators. The obtained results show accuracy and recall of the best trained model are 0.999 and 0.997, respectively. While (Shafiq et al., 2020b) obtained higher metrics score our models come close and have better performance or equal to other comparable works. The training time of the algorithms has been reduced at least 60 times (if comparing the RF implementation to Alsamiri and Alsubhi, 2019) or more. The drastic decrease in training and prediction time makes the model more feasible for deployment in the industry allowing frequent retraining sessions and quick prediction service. Application of permutation importance together with oversampling proved vital for the final improvement of both time and accuracy of the models. The final results show the significance of the data processing methods applied. Appropriate selection of dataset, its discovery and implementation of feature engineering shows that our approach is promising and in future can be tested on other IoT botnet benchmarks. We offered improvements of both training and prediction times in comparison to other works in the field, while retaining high accuracy and robustness of the models.

It is important to emphasise the role of hardware for this project. The introduction of GPU for machine learning gives new possibilities allowing to solve issues that CPU cannot handle in a reasonable time. Knowing the performance of algorithms utilizing GPU the future work may involve training on larger set of data. Larger input may allow model to learn more information about the problem and as a result perform better. The future work can also involve the generation of a dataset with a larger number of minority class samples (DDoS HTTP, DDoS, TCP, DDoS UDP, DoS HTTP, DoS UDP, DoS TCP) to avoid the introduction of synthetic data which while helpful can never represent a real-life data. Other publicly available datasets could also be considered, however various datasets consist different attacks which means abundance of some classes that were used in this research. In many cases PCAP files are often available, thus future research may involve extraction of features that Koroniotis et al. (2019) used in the IoT-Botnet set.

Funding

This work was supported by research incentive funds (R20090) and Provost Research Fellowship Grant (R20093), Zayed university, United Arab Emirates.

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.

CRediT authorship contribution statement

Michal Motylinski: Conceptualization, Data curation, Methodology, Resources, Formal analysis, Writing – original draft, Writing – review & editing. **Áine MacDermott:** Conceptualization, Data curation, Resources, Methodology, Project administration, Supervision, Writing – original draft, Writing – review & editing. **Farkhund Iqbal:** Conceptualization, Project administration, Supervision, Funding acquisition, Writing – review & editing. **Babar Shah:** Conceptualization, Project administration, Funding acquisition, Writing – review & editing.

Data availability

We have used a public dataset and referenced it within the paper.

References

Abdiansah, A., Wardoyo, R., 2015. Time complexity analysis of support vector machines (SVM) in LibSVM. Int. J. Comput. Appl. 128 (3), 28–34.

Alsamiri, J., Alsubhi, K., 2019. Internet of Things cyber-attacks detection using machine learning. Int. J. Adv. Comput. Sci. Appl 10 (12), 627–634. Available: www.ijacsa.thesai.org .

Altman, N.S., 1991. BU-1065MA An Introduction to Kernel and Nearest Neighbor Nonparametric Regression An Introduction to Kernel and Nearest Neighbor Nonparametric Regression. Cornell University 1991.

Azwar, H., Murtaz, M., Siddique, M., Rehman, S., 2018. Intrusion detection in secure network for cybersecurity systems using machine learning and data mining. In: Proceedings of the IEEE 5th International Conference on Engineering Technologies and Applied Sciences (ICETAS). IEEE, pp. 1–9. doi:10.1109/ICETAS.2018.8629197.

Breiman, L., 2001. Random forests. Mach. Learn. 45 (1), 5–32. doi:10.1023/A:1010933404324.

Churcher, A., Ullah, R., Ahmad, J., Masood, F., Gogate, M., Alqahtani, F., Buchanan, W.J., 2021. An experimental analysis of attack classification using machine learning in iot networks. Sensors 21 (2), 446. doi:10.3390/s21020446.

cuML (2022). *Welcome to cuML's documentation! — Cuml 21.06.00 documentation.*Retrieved from https://docs.rapids.ai/api/cuml/stable/. Accessed August 08, 2022.

Garre, J.T.M., Pérez, M.G., Ruiz-Martínez, A.R., 2021. A novel machine learning-based approach for the detection of SSH botnet infection. Future Gener. Comput. Syst. 115, 387–396. doi:10.1016/j.future.2020.09.004.

G Data. (2022) *Cyber-attacks on Android devices on the rise.*Retrieved from https://www.gdatasoftware.com/blog/2018/11/31255-cyber-attacks-on-android-devices-on-the-rise Accessed August 09, 2022.

Kaspersky. (2022) *Top 7 mobile security threats in 2021.* Retrieved from https://www.kaspersky.com/resource-center/threats/top-seven-mobile-security-threats-smart-phones-tablets-and-mobile-internet-devices-what-the-future-has-in-store?campaign=tcid_admitad_6ab325772c71d0e99b5c5a2683dc3a2e_240682_x4&ADDITIONAL_reseller=tcid_adm. Accessed June 30, 2022.

Koroniotis, N., Moustafa, N., Sitnikova, E., Turnbull, B., 2019. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-IoT dataset. Future Gener. Comput. Syst. 100, 779–796.

Koroniotis, N., Moustafa, N., Sitnikova, E., Slay, J., 2017. Towards developing network forensic mechanism for botnet activities in the IoT based on machine learning techniques. In: Proceedings of the International Conference on Mobile Networks and Management. Springer, Cham, pp. 30–44.

Koroniotis, N., Moustafa, N., 2020. "Enhancing network forensics with particle swarm and deep learning: the particle deep framework." 8, 209802-209834, arXiv preprint arXiv:2005.00722.

Javed, A.R., Jalil, Z., Moqurrab, S.A., Abbas, S., Liu, X., 2020. Ensemble Adaboost classifier for accurate and fast detection of botnet attacks in connected vehicles. Trans. Emerg. Telecommun. Technol. 2020. doi:10.1002/ett.4088.

Medium (2021). *Do we really need GPU for deep learning? - CPU vs GPU*. Retrieved from https://medium.com/@shachishah.ce/do-we-really-need-gpu-for-deep-learning-47042c02efe2. Accessed June 17, 2021.

Miller, S., Busby-Earle, C., 2016. The role of machine learning in botnet detection. In: Proceedings of the 11th International Conference for Internet Technology and Secured Transactions (ICITST), pp. 359–364. doi:10.1109/ICITST.2016.7856730.

Nanni, L., Brahnam, S., Ghidoni, S., Lumini, A., 2015. Toward a general-purpose heterogeneous ensemble for pattern classification. Comput. Intell. Neurosci. 2015. doi:10.1155/2015/909123.

Oreški, D., Andročec, D., 2020. Genetic algorithm and artificial neural network for network forensic analytics. In: Proceedings of the 43rd International Convention on Information, Communication and Electronic Technology (MIPRO). IEEE, pp. 1200–1205. doi:10.23919/MIPRO48935.2020.9245140.

Pohar, M., Blas, M., Turk, S., 2004. Comparison of logistic regression and linear discriminant analysis: a simulation study. Metodol. Zv. 1 (1), 143.

Shafiq, M., Tian, Z., Bashir, A.K., Du, X., Guizani, M., 2020a. CorrAUC: a malicious bot-IoT traffic detection method in IoT network using machine-learning techniques. IEEE Int. Things J. 8 (5), 3242–3254. doi:10.1109/jiot.2020.3002255.

Shafiq, M., Tian, Z., Sun, Y., Du, X., Guizani, M., 2020b. Selection of effective machine learning algorithm and Bot-IoT attacks traffic identification for internet of things in smart city. Futur. Gener. Comput. Syst. 107, 433–442. doi:10.1016/j.future.2020.02.017.

Sujatha, P., Mahalakshmi, K., 2020. Performance evaluation of supervised machine learning algorithms in prediction of heart disease. In: Proceedings of the IEEE International Conference for Innovation in Technology (INOCON). IEEE, pp. 1–7. doi:10.1109/INOCON50539.2020.9298354.

Threatpost. (2022) D-*link, IoT devices under attack by Tor-based Gafgyt variant*. Retrieved from https://threatpost.com/d-link-iot-tor-gafgyt-variant/164529/. Accessed July 28, 2022.

UNSW Canberra (2022) *The Bot-IoT dataset*. Retrieved from https://research.unsw.edu.au/projects/bot-iot-dataset. Accessed June 08, 2022.

Vakili, M., Ghamsari, M., & Rezaei, M. (2020). Performance analysis and comparison of machine and deep learning algorithms for IoT data classification. arXiv preprint arXiv:2001.09636.

Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., Wang, C., 2018. Machine learning and deep learning methods for cybersecurity. IEEE Access 6, 35365–35381. doi:10.1109/ACCESS.2018.2836950.

Zhang, C., Liu, C., Zhang, X., Almpanidis, G., 2017. An up-to-date comparison of state-of-the-art classification algorithms. Expert Syst. Appl. 82, 128–150. doi:10.1016/j.eswa.2017.04.003.

Mr Michał Motyliński is a data scientist working in collaboration between the School of Computer Science and Mathematics at Liverpool John Moores University (LJMU) and Zayed University, UAE. Michal has an MSc Artificial Intelligence (Machine Learning) graded distinction from Liverpool John Moores University (2022), and a first-class BSc (Hons) in Computer Forensics. He is currently interested in research centred around the application of machine learning for digital forensics and cybersecurity.

Dr Áine MacDermott is a Senior Lecturer in the School of Computer Science and Mathematics at Liverpool John Moores University (LJMU) in the UK. Additionally, Dr MacDermott is an Adjunct Professor in the Faculty of Business and IT, at Ontario Tech University, Canada. This role is in recognition of Áine's knowledge, skills and expertise in digital forensics and cyber security. At LJMU, Áine teaches on both the Computer Forensics and Computer Security programmes. She obtained her PhD in Network Security from LJMU in 2017, and a BSc (Hons) in Computer Forensics in 2011. Áine is also member of Research Centre for Critical Infrastructure Computer Technology and Protection (PROTECT) at LJMU, with research interests including the Internet of Things, collaborative intrusion detection in interconnected networks, digital forensics, and machine learning.

Dr Farkhund Iqbal holds the position of Associate Professor and Director Advanced Cyber Forensics Research Laboratory in the College of Technological Innovation, Zayed University, United Arab Emirates. He is leading Cybersecurity and Digital Forensics (CAD) research group in center for Smart Cities and Intelligent Systems, Zayed University. He holds a Master (2005) and a Ph.D. degree (2011) from Concordia University, Canada. He is using Artificial Intelligence, Machine Learning and Data Analytics for problem-solving in health care, cybersecurity, and cybercrime investigation in smart and safe city domain. He has published more than 100 papers in high ranked journals and conferences. He is an Affiliate Professor in the School of Information Studies, McGill University, Canada, and an Adjunct Professor in the Faculty of Business and IT, University of Ontario Institute of Technology, Canada. He has served as a chair and TPC member of several IEEE/ACM conferences and is the reviewer for several high-rank journals. He also chairs Cyberbercrime Investigation and Digital Forensics workshop and Computing in Companion Robots and Smart Toys Symposium in the Hawaii International Conference on System Sciences (HICSS).

Dr. Babar Shah is an Associate Professor at the College of Technological Innovation, Zayed University, UAE. Dr. Babar received MS degree (2007) from University of Derby, UK and PhD (2014) from Gyeongsang National University, South Korea. Dr. Babar professional services includes but are not limited to - Guest Editorships, University Services, Workshops Chair, Technical Program Committee Member, and reviewer for several international journals and conferences. His research work is associated with interdisciplinary field of Network Sciences, Health Informatics, Information Systems, Machine and Deep Learning.